Universidad Nacional Mayor de San Marcos
Universidad del Perú. Decana de América
Facultad de Ingeniería de Sistemas e Informática
Escuela Profesional de Ingeniería de Software

Definición e implementación del proceso de pruebas de software basado en la NTP-ISO/IEC 12207:2016 aplicado a una empresa consultora de software

TESIS

Para optar el Título Profesional de Ingeniero de Software

AUTOR

Evelyn Joanna CHINARRO MORALES

ASESOR

María Elena RUIZ RIVERA

Lima, Perú

2019

Referencia bibliográfica

Chinarro, E. (2019). *Definición e implementación del proceso de pruebas de software basado en la NTP-ISO/IEC 12207:2016 aplicado a una empresa consultora de software*. Tesis para optar el título de Ingeniero de Software. Escuela Profesional de Ingeniería de Software, Facultad de Sistemas e Informática, Universidad Nacional Mayor de San Marcos, Lima, Perú.

METADATOS COMPLEMENTARIOS

1. Código ORCID del autor: 0000-0001-7077-5117.
2. Código ORCID del asesor: 0000-0003-3300-7068.
3. DNI del Autor: 71796666.
4. Grupo de Investigación: Ingeniería De Software Basado En Algoritmos Para Buscar Información En Grandes Base De Datos.
5. Institución que financia parcial o totalmente la investigación:

   Universidad Nacional mayor de San Marcos.
6. Ubicación Geográfica donde se desarrolló la investigación:

   Instalaciones de la Universidad nacional Mayor de San Marcos:

   Calle Germán Amézaga N° 375 - Edificio Jorge Basadre, Ciudad Universitaria, Lima 1.
   Latitud: -10 || Longitud: -75.25
7. Año o rango de años que la investigación abarcó:

   Año y medio.

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
Universidad del Perú, DECANA DE AMÉRICA
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
ESCUELA PROFESIONAL DE INGENIERÍA DE SOFTWARE

Acta de Sustentación de Tesis

Siendo las 16:10 del día 12 de febrero del año 2019, se reunieron los docentes designados como miembros de Jurado de la Tesis, presidido por el Ing. Fany Yexenia Sobero Rodríguez, Lic. Víctor Hugo Bustamante Olivera (Miembro), y la Lic. María Elena Ruiz Rivera (Miembro Asesor) para la sustentación de la Tesis intitulada: **"DEFINICIÓN E IMPLEMENTACIÓN DEL PROCESO DE PRUEBAS DE SOFTWARE BASADO EN LA NTP-ISO/IEC 12207: 2016 APLICADO A UNA EMPRESA CONSULTORA DE SOFTWARE "**; por la bachiller **Evelyn Joanna Chinarro Morales**, para optar el Título Profesional de Ingeniero de Software.

Acto seguido de la exposición de la Tesis, el Presidente invitó a la bachiller a dar respuesta a las preguntas establecidas por los Miembros de Jurado.

La bachiller en el curso de sus intervenciones demostró pleno dominio del tema, al responder con acierto y fluidez a las observaciones y preguntas formuladas por los señores miembros del Jurado.

Finalmente habiéndose efectuado la calificación correspondiente por los miembros de Jurado, la bachiller obtuvo la nota de 16.. (En letras)..Dieciseis.....

A continuación el Presidente del Jurado, Ing. declara a la Bachiller **Ingeniero de Software**.

Siendo las 16:40 .... horas, se levantó la sesión.

.....
Ing. Fany Yexenia Sobero Rodríguez
Presidente

.....
Lic. Víctor Hugo Bustamante Olivera
Miembro

.....
Lic. María Elena Ruiz Rivera
Miembro Asesor

FICHA CATALOGRÁFICA

Chinarro Morales, Evelyn

DEFINICIÓN E IMPLEMENTACIÓN DEL PROCESO DE PRUEBAS DE SOFTWARE BASADO EN LA NTP-ISO/IEC 12207:2016 APLICADO A UNA EMPRESA CONSULTORA DE SOFTWARE

Tesis, Pregrado: “Universidad Nacional Mayor de San Marcos – Facultad de Ingeniería de Sistemas e Informática – Escuela Profesional de Ingeniería de Software”

Formato 28 x 20 cm Páginas #111

DEDICATORIA:

Este trabajo está dedicado a mi familia, pero en especial mi madre que siempre ha estado apoyándome y velando por mi durante toda mi vida.

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

ESCUELA PROFESIONAL DE INGENIERÍA DE SOFTWARE

Definición e Implementación Del Proceso De Pruebas De Software Basado en la NTP-ISO/IEC 12207:2016 Aplicado a una Empresa Consultora De Software

Autor: **Chinarro Morales, Evelyn Joanna**

Asesor: **Ruiz Rivera, María Elena**

Título: **Tesis para optar el Título Profesional de Ingeniero de Software**

Fecha: **Febrero 2019**

RESUMEN

Esta tesis se enfoca en definir e implementar un proceso para realizar las pruebas funcionales basado en herramientas ya creadas para su automatización durante la evaluación de la calidad del producto desarrollado. Esta automatización contempla la comparación de ciertas herramientas para la administración de pruebas funcionales.

El desarrollo del software y el brindar un servicio informático de primera calidad es lo que actualmente se logra con las pruebas funcionales hechos en la etapa de desarrollo. Este proceso es parte muy importante y crítica dentro del proceso de desarrollo, debiendo realizarse con la mayor eficacia y la mejor eficiencia.

Durante la tesis se tenía conocimiento sobre los procesos actuales que forman parte del proceso de desarrollo, tanto teórico como en la práctica. Se considera muy importante que las soluciones técnicas apoyen diferentes aspectos como los siguientes: encontrar defectos en fases iniciales del desarrollo, lograr mejor cobertura de la funcionalidad durante las pruebas y la ejecución de estas en costo y tiempo.

Palabras claves: pruebas de software, Casos de Prueba, Automatización de pruebas.

MAJOR NATIONAL UNIVERSITY OF SAN MARCOS

FACULTY OF SYSTEMS AND COMPUTER ENGINEERING

PROFESSIONAL SCHOOL OF SOFTWARE ENGINEERING

Definition and Implementation of the Software Testing Process Based on the NTP-ISO / IEC 12207: 2016 Applied to a Software Consulting Company

Author: **Chinarro Morales, Evelyn Joanna**

Advisor: **Ruiz Rivera, María Elena**

Title: **Thesis to choose the Professional Title of Software Engineer**

Date: **February 2019**

ABSTRACT

This thesis focuses on defining and implementing a process to perform functional tests based on tools already created for automation during the evaluation of the quality of the product developed. This automation contemplates the comparison of certain tools for the administration of functional tests.

The development of software and the provision of a high quality IT service is what is currently achieved with the functional tests done in the development stage. This process is a very important and critical part of the development process, and must be carried out with the greatest efficiency and effectiveness.

During the thesis there was knowledge about the current processes that are part of the development process, both theoretical and in practice. It is considered very important that technical solutions support different aspects such as the following: finding defects in early stages of development, achieving better coverage of the functionality during the tests and the execution of the same in cost and time.

Key Words: Software testing, Test cases, Test Automation.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La presente tesis consiste en definir e implementar un proceso para realizar las pruebas funcionales en una empresa consultora de software ya que actualmente se realizan manualmente, esto origina que cada día los usuarios presenten diversidad de incidencias causadas por las fallas y /o errores no encontradas durante la etapa de pruebas realizadas al producto antes de pasar a producción.

Actualmente, el área de Control de Calidad de la empresa consultora realiza las pruebas funcionales de manera manual y tomando bajo criterio, el poco conocimiento que se tiene del negocio, esto ocasiona que la empresa tenga que invertir más tiempo y recursos para subsanar los errores no encontrados.

Según lo menciona Glenford J. Myers: “El probador de software necesita la actitud adecuada (quizás "visión" es una palabra mejor) para probar con éxito una aplicación de software. En algunos casos, la actitud del evaluador puede ser más importante que el proceso en sí” [3]. Este punto denota que cuando se tiene la visión del proceso del negocio que se está evaluando y las características que contiene, el evaluador enfatiza las pruebas en los puntos más críticos para el negocio.

Es así como la empresa consultora de software manifiesta el interés de mejorar su proceso de pruebas funcionales y adicionalmente utilizar y apoyarse en el uso de herramientas para automatización de pruebas funcionales para así optimizar el tiempo dedicado a las pruebas y garantizar la entrega de un producto de calidad a sus usuarios. Según lo menciona la IEEE: “La calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario” [13], punto esencial que la empresa consultora desea mejorar y darle valor al producto de software que ellos desarrolla para sus usuarios finales.

El presente documento está organizado en capítulos. Al inicio encontrará un resumen de la tesis y la introducción, en el capítulo I se encuentra el planteamiento metodológico, así como los objetivos específicos además del problema y la justificación para la realización de ésta tesis.

Luego en el capítulo 2, se encuentra el estado del arte en pruebas funcionales manuales y automatizadas, el cuál trata desde temas más generales, como modelos de madurez de los procesos de pruebas hasta las herramientas de apoyo que se utilizan para realizar las pruebas funcionales.

CAPÍTULO I. PLANTEAMIENTO METOLÓGICO

1.1 Antecedentes del Problema

Los sistemas o productos de software son creados por seres humanos por ende no se puede garantizar que no se hayan cometido errores durante la implementación de estos.

Asimismo, el costo asumido por los errores encontrados en el sistema, luego de haberse adquirido, suponía mucha pérdida de dinero y tiempo de los recursos asignados para el proceso de desarrollo, es así que se inició con las pruebas de software para garantizar la disminución de fallos del sistema en producción.

Anteriormente las pruebas de software se orientaban a corregir directamente el código fuente de los aplicativos. Éstas eran realizadas directamente por los programadores.

Según lo menciona el autor Myers: "Desde un punto de vista psicológico, el análisis y diseño de software (junto con la codificación) son tareas constructivas. El ingeniero de software analiza, modela y luego crea un programa de computadora y su documentación. Como cualquier constructor, el ingeniero de software está orgulloso del edificio que construyó y ve con desconfianza a quien intente derrumbarlo.

Cuando comienzan las pruebas, hay un sutil, pero definitivo, intento por "romper" lo que construyó el ingeniero de software. Desde el punto de vista del desarrollador, las pruebas pueden considerarse como (psicológicamente) destructivas. De modo que el desarrollador actuará con cuidado, y diseñará y ejecutará pruebas que demostrarán que el programa funciona, en lugar de descubrir errores. Desafortunadamente, los errores estarán presentes. Y si el probador no los encuentra, ¡el cliente lo hará!" [3].

Conforme se ha ido avanzando en el tiempo y en el avance de la tecnología se inició la utilización masiva de pruebas para garantizar que se cumple con la especificación de los requerimientos dados por el usuario.

Estas pruebas realizadas se transforman en Casos de Prueba que se ejecutan a los aplicativos para encontrar errores y realizar las correcciones.

En la actualidad se propone una metodología que contempla el análisis, la revisión y pruebas en el ciclo devida del desarrollo del software. El testing empieza a integrarse en las diferentes metodologías del desarrollo. El objetivo general de las pruebas de software es confirmar y asegurar la calidad de los sistemas.

Para implantar con éxito una estrategia de prueba de software es necesario abordar los puntos siguientes según los autores Gelperia y Hetzel: " especificar los requisitos del producto de manera cuantificable mucho antes que comiencen las pruebas, establecer los objetivos de la prueba de manera explícita, comprender que usuarios van a manejar el software y desarrollar un perfil para cada categoría de usuario, desarrollar un plan de pruebas, llevar a cabo revisiones técnicas formales para evaluar la estrategia de prueba y los propios casos de prueba " [1].

Existen diferentes tipos de pruebas de software asociadas a técnicas específicas en el cual se diseñarán, ejecutarán y evaluarán teniendo como referencia la estructura interna del aplicativo. Las principales técnicas son: Caja Blanca, Caja Gris, Caja Negra, Pruebas de Unidad y las Pruebas del Sistema, que está constituida por una serie de pruebas diferentes (prueba de recuperación, prueba de seguridad, prueba de resistencia y prueba de rendimiento).

Se debe tener en cuenta que las organizaciones que están inmersas en cualquier proceso del ciclo de vida de software realizan evaluaciones de los productos. Los procesos de Verificación de Software y Validación de Software brindan la oportunidad para evaluaciones adicionales. Estos procesos son conducidos por el proveedor o una parte independiente con el fin de verificar y validar los productos, con profundidad variable dependiendo del proyecto. [11]

Dado lo expuesto, se denota que la etapa perteneciente a las pruebas de software es una parte fundamental en el proceso de desarrollo de software así que se debe tener en consideración un tiempo definido para su ejecución además de definir el proceso a seguir

y apoyarse en herramientas para así asegurar la óptima ejecución de las pruebas funcionales.

1.2. Definición del Problema

Actualmente, el área de Calidad de la empresa consultora de software realiza sus pruebas funcionales de manera manual y tomando bajo criterio, el poco conocimiento que se tiene del negocio, esto ocasiona que la empresa tenga que invertir más tiempo y recursos para subsanar los errores no encontrados, al mismo tiempo ocasiona que los usuarios reporten incidencias con frecuencia las cuales retrasa el óptimo desarrollo de sus funciones.

1.3. Objetivos

1.3.1 Objetivo Principal

Desarrollar e implementar un proceso para la realización de las pruebas funcionales de software basada en la NTP-ISO/IEC 12207:2016.

1.3.2 Objetivos Secundarios

- Definir las actividades que se seguirán en el proceso planteado basado en la NTP-ISO/IEC 12207:2016 de acuerdo con el principio de Verificación y Validación.
- Documentar todos los procesos del negocio para facilitar la definición de los Casos de Prueba durante el desarrollo del aplicativo.
- Evaluar y seleccionar una herramienta de apoyo para la automatización de las pruebas funcionales de software y así garantizar una mejora en el proceso.
- Disminuir el tiempo de ejecución de las pruebas realizadas en la organización.

1.4. Justificación

Actualmente la realización de la ejecución de las pruebas funcionales en la empresa consultora se realiza manualmente, registrando las incidencias encontradas en un documento Excel. Se mapean los bugs y sólo se prueba funcionalmente los flujos conocidos del negocio, esto causa que constantemente el usuario reporte diversos errores por no contar con un proceso en cual se basen para realizar las pruebas funcionales de manera óptima y así brindar un producto con mejor calidad.

Basándonos en el descrito por Glenford J. Myers en su libro "The Art of Testing Software" el cual manifiesta que: "cuando se prueba un programa se desea agregarle algún valor (es decir, puesto que la prueba es una actividad costosa, se desea recuperar parte de este costo por medio de un incremento del valor del programa). Agregar valor significa aumentar su calidad o confiabilidad, lo que, a su vez, significa encontrar y eliminar errores". [3]

Atendiendo la necesidad de la empresa consultora se desarrolló este proceso el cual beneficiará a la empresa a mejorar su proceso de pruebas de software, ahorrando tiempo y recursos además de garantizar el buen funcionamiento del aplicativo ya que en la etapa de desarrollo se mapearán la mayor cantidad de errores que se puedan encontrar al realizar las pruebas de los aplicativos desarrollados.

Asimismo, se utilizará una herramienta de apoyo existente, ya que se necesita para automatizar las pruebas funcionales, con la finalidad que éstas se puedan ejecutar de forma rápida y automática.

Según lo manifiesta Pressman: "se debe tener en cuenta que las pruebas representan el último bastión desde donde puede valorarse la calidad y, de manera más pragmática, descubrirse errores. Pero las pruebas no deben verse como una red de seguridad. Como se dice: no se puede probar la calidad. Si no está ahí antes de comenzar las pruebas, no estará cuando termine de probar. La calidad se incorpora en el software a lo largo de todo el proceso de ingeniería del software. La adecuada aplicación de métodos y herramientas, revisiones técnicas efectivas, y gestión y medición sólidas conducen a la calidad que se confirma durante las pruebas". [2]

1.5 Propuesta Metodológica

La solución involucra la definición e implementación de un nuevo proceso para la realización de las pruebas funcionales a los aplicativos antes de ser entregados al cliente.

Adicionalmente nos apoyaremos en el uso de una herramienta para automatizar las pruebas funcionales del aplicativo y obtener la mayor cantidad de defectos, también guardaremos un histórico de los casos de pruebas realizados y evaluados para futuros mantenimientos del aplicativo desarrollado y evaluado en la presente tesis.

1.6 Organización de la tesis

La presentación de la tesis está organizada de la siguiente forma:

En el Capítulo II, se describirá brevemente los conceptos básicos sobre las pruebas de software, conceptos primordiales para la realización de los tipos de pruebas funcionales que se realizan, así como las herramientas de apoyo que se utilizan para realizar las pruebas funcionales del aplicativo, para un mejor entendimiento de la presente tesis.

En el Capítulo III se abordará el estado de arte, con el análisis de las diferentes metodologías implementadas por autores que han estado investigando sobre el tema, así como información primordial sobre las experiencias adquiridas durante el desarrollo de los artículos.

En el Capítulo IV, se implementará el proceso y se describirán las actividades que se desarrollarán en el basándose en la norma NTP-ISO/IEC 12207:2016.

En el Capítulo V, se colocará el aporte práctico, implementación del proceso definido a un aplicativo de la empresa consultora.

En el Capítulo VI, se presentarán las conclusiones y recomendaciones de este trabajo.

Finalmente, presentaré las referencias bibliográficas que me ha servido para la realización de la presente tesis.

CAPITULO II. MARCO TEÓRICO

2.1. Conceptos Básicos de Pruebas de Software

2.1.1. Base Teórica de las Pruebas. [WEB01]

Las pruebas se clasifican mediante tipos de pruebas y estos tipos están relacionados con respecto a la función que cumplen. Para su clasificación se toman en cuenta tanto cómo funcionan las pruebas como también lo que estas prueban. Es importante tener clara la clasificación ya que con esto tendremos un mejor conocimiento de la funcionalidad, es decir para que se utilizan y cuando emplear un tipo de prueba especifico.

2.1.2. Gestión de Pruebas [WEB02]

Para el desarrollo de pruebas de software no solo hace falta el conocimiento de cómo desarrollarlas, también se deben realizar otro conjunto de tareas; entre estas tareas encontramos la planificación de pruebas, la generación de casos de uso, el seguimiento de los errores, manejar las configuraciones del sistema y otras tareas. La realización de todas esas tareas es lo que se denomina gestión de pruebas dado que la mayoría de las veces son supervisadas por un individuo y realizadas por un grupo de trabajo. Para el caso de este trabajo se utilizarán las tareas de gestión de errores, generación de casos de uso, planificación de las pruebas y el manejar configuraciones para establecer que una herramienta automatizada da apoyo en la gestión de pruebas de software.

2.1.3. Pruebas funcionales [3]

Encontramos la definición en el libro de Myers: “Las pruebas funcionales son un proceso que intenta encontrar discrepancias entre el programa y la especificación externa. Una especificación externa es una descripción precisa del comportamiento del programa desde el punto de vista del usuario final.

Excepto cuando se usa en programas pequeños, las pruebas funcionales es normalmente una actividad de caja negra. Es decir, confía en el proceso de prueba de módulo anterior para lograr los criterios de cobertura lógica de caja blanca deseados”. [3]

Para realizar una prueba funcional, la especificación se analiza para derivar un conjunto de casos de prueba. Las pruebas son los procesos de ejecución de un programa en los que se intentan descubrir errores. Esto supone un cambio de punto de vista, nos cambia la idea

que tenemos de que una prueba es exitosa si no consigue errores. Para poder diseñar casos de prueba que proporcionen un buen resultado, el ingeniero debe tener en cuenta los principios de las pruebas.

2.1.4. Prueba Caja Blanca [3]

La prueba de caja blanca o lógica, le permite examinar la estructura interna del programa. Esta estrategia deriva los datos de prueba a partir de un examen de la lógica del programa (y, a menudo, desafortunadamente, por negligencia de la especificación).

El objetivo en este punto es establecer, para esta estrategia, la prueba de entrada analógica a exhaustiva en el enfoque de caja negra. Hacer que todas las declaraciones en el programa se ejecuten al menos una vez puede parecer la respuesta, pero no es difícil demostrar que esto es altamente inadecuado. El análogo generalmente se considera una prueba de ruta exhaustiva. Es decir, si ejecuta, a través de casos de prueba, todas las rutas posibles de control fluyen a través del programa, posiblemente el programa haya sido completamente probado. [3]

2.1.5. Prueba Caja Negra [11,2]

Una estrategia de prueba importante es la prueba de caja negra, basada en datos o de entrada / salida. Para usar este método, vea el programa como una caja negra. Su objetivo es estar completamente despreocupado por el comportamiento interno y la estructura del programa. En su lugar, concéntrese en encontrar circunstancias en las que el programa no se comporte de acuerdo con sus especificaciones.

En este enfoque, los datos de prueba se derivan únicamente de las especificaciones (es decir, sin aprovechar el conocimiento de la estructura interna del programa) [11].

Pressman indica que en este tipo nos encontramos con: la prueba basada en grafos, la partición equivalente, siendo esta última un método que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba; también nos encontramos con el Análisis de Valores Límites, la prueba de comparación y la prueba de tabla ortogonal [2]

2.1.6. Prueba de Sistema [3]

Esta fase de prueba del sistema se da cuando el sistema ya está instalado. Según el libro de Myers menciona: "Las pruebas de sistema están constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema basado en computadora. Aunque cada prueba tenga un propósito diferente, todas trabajan para verificar que se han integrado adecuadamente todos los elementos del sistema, y que realizan las funciones apropiadas. Las pruebas que se realizan en este tipo son: de Recuperación, la cual es una prueba del sistema que fuerza al fallo del software de muchas formas y verifica que la recuperación se lleva a cabo apropiadamente; de Seguridad, ésta intenta verificar que los mecanismos de protección incorporados en el sistema lo protegerán, de hecho, de accesos impropios; de Resistencia, la cual ejecuta un sistema de forma que demande recursos en cantidad, frecuencia o volúmenes anormales; y finalmente la prueba de Rendimiento, como su nombre lo indica, está diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado". [3]

2.1.7 Norma Técnica Peruana NTP-ISO/IEC 1220:2016[11]

Esta Norma establece un marco común para los procesos del ciclo de vida del software, con la terminología bien definida, que puede ser referenciada por la industria del software. En la norma desarrollada por la INACAL mencionan: "Se aplica a la adquisición de sistemas y productos y servicios software, al suministro, desarrollo, operación, mantenimiento y retiro de los productos software y la parte software de un sistema, ya sea ejecutado interna o externamente a una organización. Esos aspectos de la definición del sistema necesarios para proporcionar el contexto para los productos y servicios software, están incluidos. El software incluye la parte software del firmware. Esta revisión integra la norma ISO/IEC 12207:1995, con sus dos enmiendas y fue coordinada con la revisión paralela de la ISO/IEC 15288:2002 (procesos del ciclo de vida del Sistema) para alinear la estructura, términos y los correspondientes procesos organizativos y de proyecto". [11]

Esta Norma, dada por la Dirección de Normalización, describe que se puede utilizar en uno o más de los siguientes modos:

- "Por una organización: para ayudar a establecer un entorno de procesos deseados. Estos procesos pueden ser soportados por una infraestructura de métodos, procedimientos, técnicas, herramientas y personal capacitado. La organización puede emplear este medio para realizar y gestionar sus proyectos y el progreso de sus sistemas a través de las fases de su ciclo de vida. En este modo, esta Norma se utiliza para evaluar la conformidad de un conjunto declarado y establecido de procesos del ciclo de vida para su disposición.
- Por un proyecto: para ayudar a seleccionar, estructurar y emplear los elementos de un conjunto establecido de procesos del ciclo de vida para proporcionar productos y servicios. En este modo, Esta Norma se utiliza en la evaluación de la conformidad del proyecto para el entorno declarado y establecido". [11]

2.2. Herramientas para la Automatización de Pruebas Funcionales [WEB02]

Como ya sabemos que la etapa de pruebas puede llegar a requerir una cantidad de tiempo y esfuerzo incluso mayor que el desarrollo del software; tenemos que en la actualidad se encuentran disponibles diferentes tipos de herramientas para automatizar las pruebas que sirven de apoyo a la realización de dichas pruebas. Cuando se habla de herramientas que sirven de soporte o apoyo se habla de herramientas que permiten el desarrollo de pruebas automatizadas y el diseño de pruebas, tomando en cuenta esta funcionalidad de una herramienta estas pueden clasificarse en herramientas de apoyo a la gestión de pruebas y/o al desarrollo de pruebas de software.

Actualmente existen herramientas que ofrecen soporte en ambos campos por lo tanto se pueden tener herramientas que ofrezcan soporte completo para la etapa de pruebas en el desarrollo de software, esto dependiendo si las pruebas a las cuales dicho software da apoyo son las mismas que en el momento dado se necesitan aplicar. Un ejemplo de una herramienta de apoyo tanto para el desarrollo como para la gestión de pruebas es la herramienta JUnit, la cual proporciona soporte para las pruebas unitarias en el ambiente de desarrollo de software. Tenemos diferentes herramientas que listamos a continuación:

2.2.1. Selenium [WEB02]

"Selenium es un conjunto de utilidades que facilita la labor de obtener juegos de pruebas para aplicaciones web. Para ello nos permite grabar, editar y depurar casos de prueba, que podrán ser ejecutados de forma automática e iterativa posteriormente.

Además de ser una herramienta para registrar acciones, permite editarlas manualmente o crearlas desde cero. Las acciones se basan en el uso de diferentes Apis en diferentes lenguajes (PHP, Ruby, JAVA, JavaScript, etc.). Entre sus principales características podemos nombrar:

- Facilidad de registro y ejecución de los test.
- Referencia a objetos DOM en base al ID, nombre o a través de XPath.
- Autocompletado para todos los comandos.
- Las acciones pueden ser ejecutadas paso a paso.
- Herramientas de depuración y puntos de ruptura (breakpoints).
- Los test pueden ser almacenados en diferentes formatos.

El potencial de esta herramienta puede ser utilizado para la grabación de las pruebas funcionales durante la Generación de pruebas de regresión. Con este servicio se consigue obtener una batería de pruebas automatizadas que podrán ser utilizadas cuando sea necesario repetir las pruebas", según lo mencionado por William Steearing en su página sobre Automatización con Selenium.

Tipos de Herramientas de Selenium:

Selenium IDE [WEB03]

Permite crear Scripts de pruebas de manera rápida ya que trabaja como una "Grabadora" que captura todas las acciones que se hacen sobre la interfaz de usuario (una especie de macro), generando automáticamente los comandos del Script de pruebas.

Esta herramienta trabaja como un "complemento" (add-on) de Firefox, el cual permite Grabar y Reproducir de manera simple las iteraciones con el browser.

Otras Características:

- Manejo inteligente de la identificación de Campos o Elementos en la página Web, se puede usar IDs, nombres o XPath.
- Autocompleta todos los comandos de Selenium.

- Debug y Puntos de quiebre.
- Permite grabar los Scripts en formato HTML, Ruby, JUnit, entre otras.
- Soporte para el archivo "user-extensions.js". Más adelante se explicará la utilizada del archivo "user-extensions.js", el cual permite agregar funciones JavaScript personalizadas.
- Opciones para realizar validaciones automáticas del título para cada página en la que se esté navegando.

Selenio WebDriver [WEB03]

Permite crear Scripts de pruebas más robustos pues se puede desarrollar sobre varios lenguajes de Programación, entre ellos C# y Java como se ve en la figura 4.

Es de gran ayuda en las pruebas de Regresión y Re-Testing.

CAPÍTULO III: ESTADO DEL ARTE

3.1. Metodologías para Pruebas Funcionales de Software

3.1.1. Mejoramiento del Proceso de Pruebas [4]

Se definen algunas de las herramientas como se muestra en la figura 5 que se utiliza como apoyo para el desarrollo de éste artículo, las cuáles describiré a continuación:

De forma organizada comienza usando la herramienta Delfos, el cual tiene como objetivo principal administrar los requerimientos que el usuario requiere para el desarrollo del proyecto. El que la herramienta sea vía web permite realizar el levantamiento de la información de los requerimientos en la oficina del usuario o en otra área donde el usuario tenga disponibilidad.

Otra de las herramientas utilizadas es Arquímedes, es aquella que guarda un histórico de las pruebas realizadas para poder consultarlas en un futuro, así como los criterios o reglas utilizadas para dar las pruebas como válidas.

Los resultados son documentados por el equipo de trabajo y se notifican los defectos encontrados a todos los miembros del equipo para conocimiento. Asimismo, se utiliza la herramienta Cronos, la cual le permite crear actividades asociadas al plan de pruebas que se está ejecutando y la corrección de los defectos encontrados durante el desarrollo de las pruebas, dichas actividades son designadas a los miembros del equipo por el jefe de proyecto, esto permitirá tener un control de las actividades que desarrolla cada miembro del equipo.

El proceso de pruebas propuesto está basado un modelo en V (Figura 1), es decir que se trabaja en forma paralela tanto el desarrollo como los planes y casos de prueba con sus criterios de aceptación para que así al término del desarrollo se tengan listas las pruebas que van a ejecutarse.

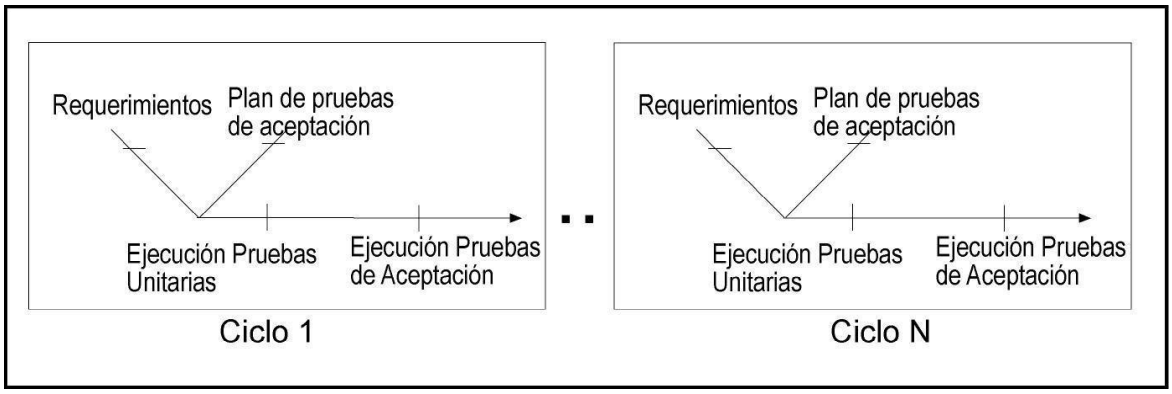


Figura 1. Modelo en V [4]

Si bien es cierto hoy en día la mejora continua de los procesos de desarrollo de software son cada día más importantes para las empresas que pretenden competir en los mercados internacionales, especialmente aquellas que cuentan con equipos de trabajo ubicados en diferentes lugares de trabajo.

Se nota el poco manejo de conocimiento y de mecanismos de coordinación entre los miembros de un equipo que puede ser realmente difícil para un proceso de desarrollo, generalmente este problema es solucionado mediante la interacción de los miembros del equipo de trabajo.

Los miembros de trabajo en un ambiente de desarrollo de software globalizado no se conocen, probablemente hablan idiomas diferentes y están en lugares diferentes, trabajan en horarios diferentes y carecen de los espacios comunes para un intercambio de información y coordinación de un proceso de pruebas efectivo.

Viendo la situación se presenta una metodología para el proceso de pruebas que se debería seguir cuando se está desarrollando un proyecto. La infraestructura consiste por un lado de las herramientas que dan apoyo a las actividades asociadas al proceso de pruebas y, por otro lado, de un mecanismo de integración de esas aplicaciones, basado en una plataforma de eventos.

El aporte se da al observar la situación de la empresa que desarrolla software y maneja distintos grupos de trabajo, así como franquicias de su empresa en distintos lugares de

ubicación, implementó una infraestructura distribuida de eventos que facilite la comunicación y coordinación entre las herramientas que se utilizarán para la comunicación entre el equipo de trabajo.

La infraestructura se llama Eleggua. Como se observa en la Figura 2, Eleggua permite la notificación y comunicación entre las herramientas de comunicación, participan en conjunto en el proceso de la generación de las pruebas y los eventos que surgieran durante el desarrollo del proyecto.

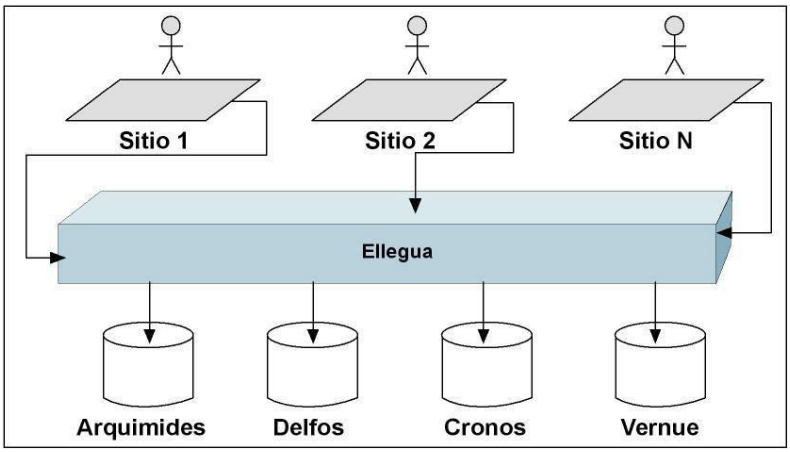


Figura 2. Infraestructura distribuida Eleggua [4]

Como se observa en la figura 3, Eleggua permite la notificación y comunicación entre las herramientas de comunicación, participan en conjunto en el proceso de la generación de las pruebas y los eventos que surgieran durante el desarrollo del proyecto.

En la figura se muestran los principales componentes de Eleggua. El middleware de eventos distribuidos (DEM), actúa como un sistema publicador/suscriptor para enviar las notificaciones de los eventos que ocurran en el proceso de desarrollo del proyecto.

Así también tenemos los proxis de cooperación que son los intermediarios entre las herramientas externas y la plataforma de eventos y finalmente permite identificar el envío y recepción de los eventos y las excepciones que puedan surgir durante el proceso.

Un Proxy de cooperación consta de tres componentes principales: componente observador interno (ofrece el servicio de observación y la generación de los eventos), componente receptor de eventos (registro de las herramientas y la notificación de los eventos) y un

componente procesador de eventos. Por último, el Monitor se encarga de controlar la plataforma de eventos.

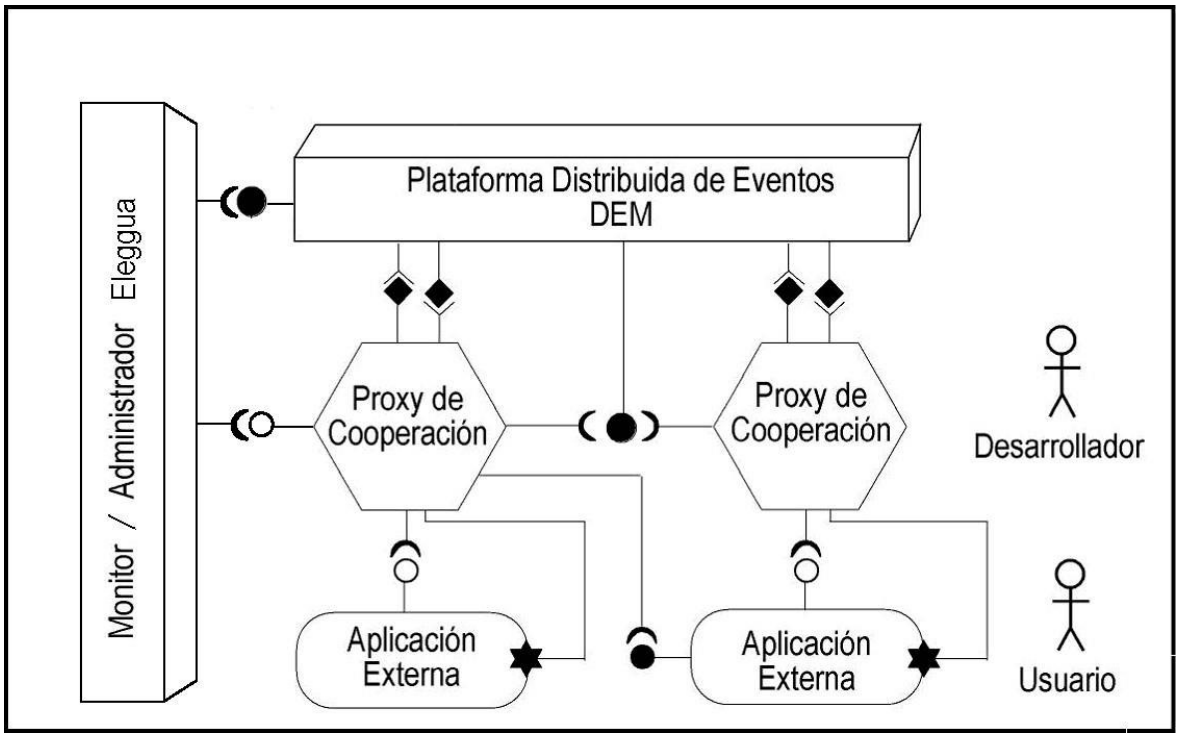


Figura 3. Componentes de la Infraestructura Distribuida Eleggua [4]

Como una mejora se encontró que la utilización del componente AspectJ que se utiliza en la infraestructura implementada como mecanismo de observación de las aplicaciones impone distintas restricciones para su correcto funcionamiento.

Se resalta que es importante resaltar el uso de esta tecnología ya que impone un reto no solo técnico sino organizacional para las empresas, es que involucra un cambio de cultura de parte de los colaboradores.

Es así que se menciona en el artículo: "En un futuro continuarán mejorando la implementación desarrollada es así que se va a trabajar en el componente de monitoreo y

administración para agregarle más funcionalidades y que brinde un mayor apoyo a los administradores de la plataforma y a los desarrolladores de la integración.

Adicionalmente se trabajará en la integración de herramientas de planeación y gestión de proyectos tales como MS-Project para que pueda existir una integración las tareas generadas en la herramienta Cronos".

3.1.2. Método para Generar Casos de Pruebas Funcionales [5]

Las pruebas de software se dividen en dos grandes grupos, en este artículo se tendrá en cuenta las funcionales, las cuales se aplican al producto final, luego de la fase de desarrollo, éstas permiten detectar cuáles son los puntos que no cumplen los requerimientos dados por el usuario además que puedan fallar en cuanto a la funcionalidad desarrollada.

Para realizar dichas pruebas se debe planificar su ejecución por tanto se debe definir los puntos que se van a verificar y que son críticos para el buen funcionamiento del aplicativo, para lo cual se definen los casos de prueba.
Se define un método para generar los casos de pruebas funcionales basándose en los casos de uso del sistema los cuales contienen el funcionamiento de este.
Para definir la metodología se analizó métodos existentes como el de Heumann, quien desarrolla un método para generar casos de prueba tomando los casos de uso como inicio y al final se genera una lista de casos de prueba con los valores y posibles resultados esperados.

Así como también se analizó la propuesta de Riebisch, la cual utiliza la descripción de los casos de uso y ampliar su cobertura ya que aumenta la precondición y la post condición, el flujo alternativo y las referencias hacia otros casos de uso si loshubiera.

Se ha descrito propuestas de metodologías para generar casos de pruebas de otros autores, que a pesar de estar bien estructuradas no facilitan la generación de los casos de prueba porque contienen demasiados pasos y modelos que se crean antes de generar los casos de prueba y que hacen engorroso el proceso de la generación de casos de pruebas y poco entendible.

No es factible ni práctico para las empresas que se dedican a realizar las pruebas de software, al contrario, dificultan el proceso y hacen que se dedique más tiempo para realizarlos.

Luego de analizar y estudiar las metodologías ya existentes y agregándole la experiencia acumulada a través del desarrollo del proyecto en la empresa donde labora la autora, le permitió desarrollar un nuevo método para crear casos de pruebas funcionales. En la Figura 4 se puede observar la nueva metodología.

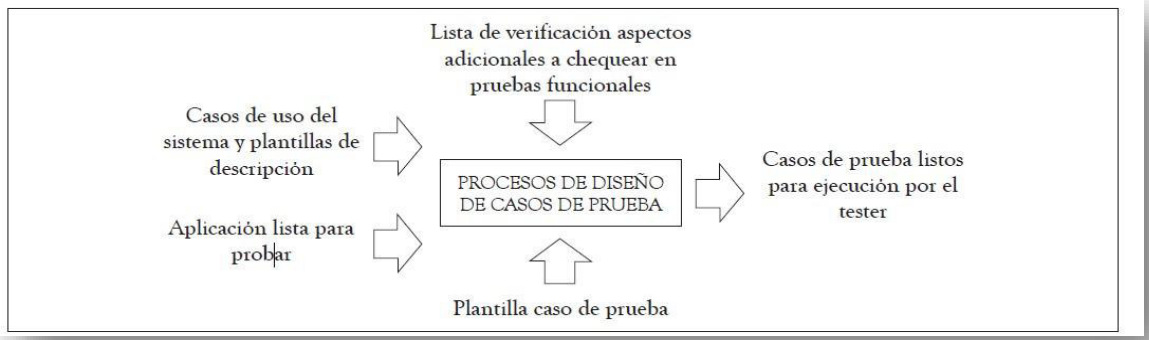


Figura 4 . Entradas y salidas de la metodología propuesta. [5]

Para generar los casos de pruebas funcionales de acuerdo con la nueva metodología primero se debe tener en cuenta la información de entrada que se necesita para poder obtener los casos de prueba a ejecutar.

Uno de los inputs es la lista de verificación de los aspectos adicionales que se deben tener en cuenta para las pruebas funcionales, esto permitirá saber si se probaron los casos

críticos para el aplicativo, otro punto son los casos de uso del sistema y las plantillas con la información y descripción de cada uno de ellos.

Es vital en este proceso contar con el aplicativo ya terminado para poder comenzar con las pruebas funcionales y verificar que se cumplan los requerimientos dados por el usuario. Asimismo, es necesario contar con una plantilla inicial de los casos de prueba a ejecutar. Como salida obtendremos los casos de pruebas funcionales que se ejecutarán en el aplicativo y que reafirmarán el buen funcionamiento de este antes de la entrega al usuario final.

En la Figura 5 se explica los pasos necesarios para el diseño de la generación de los casos de pruebas funcionales.

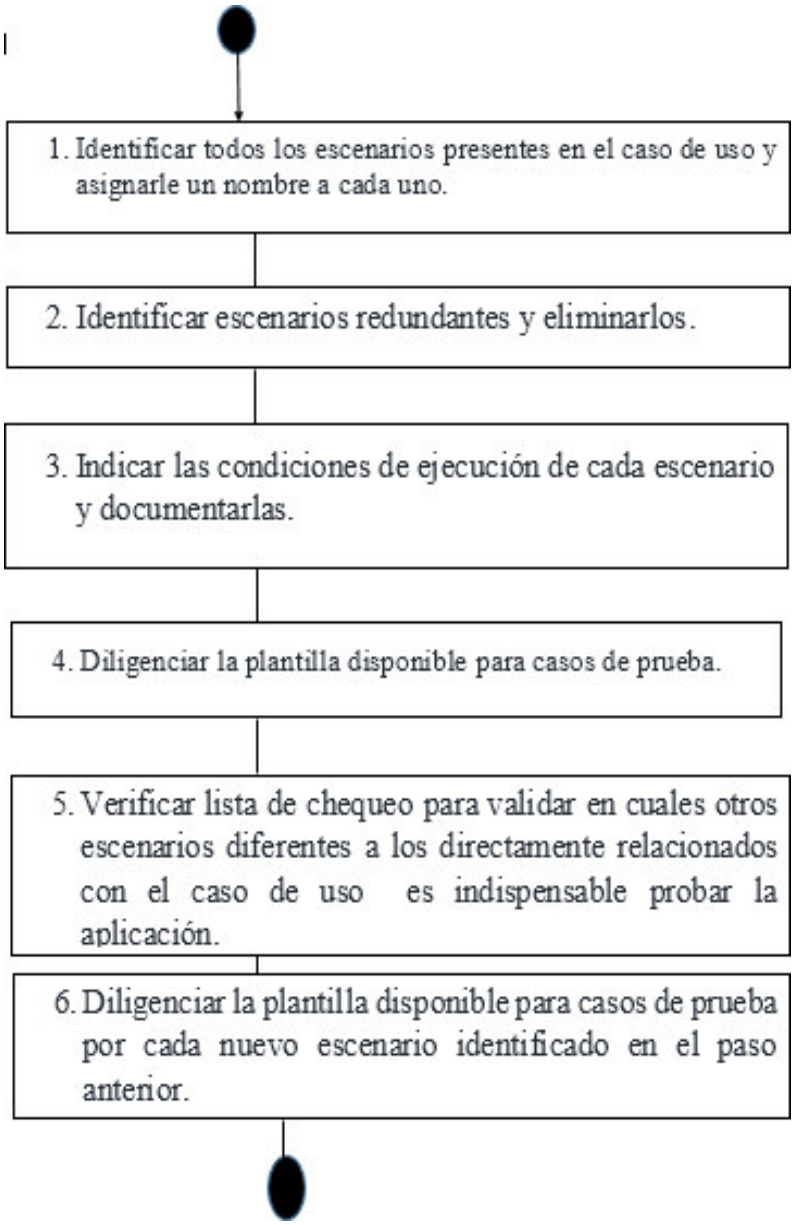


Figura 5. Proceso para creación de los casos de pruebas funcionales. [5]

Como se muestra en la Figura 5 se describe paso a paso la metodología propuesta, en el paso 1 se describe que se debe identificar los escenarios relevantes y correspondientes a los flujos normales, de error o alternativos que sucedan al ejecutar el caso de prueba.

En el paso 2 es tener en cuenta los escenarios ya identificados en el paso 1 y eliminarlos si es que se encontrarán repetidos y validados en otro escenario. En cuanto al paso 3, cada escenario que se tenga luego de realizar la eliminación de los repetidos será un caso de prueba.
En el paso 4 se hace referencia a documentar toda la información requerida en la plantilla de los casos de prueba que se están identificando. En el paso 5 de tendrá en cuenta las condiciones para ejecutar los casos de prueba, los valores necesarios para poder probar el buen funcionamiento del aplicativo por ejemplo los campos adicionales, los campos obligatorios, los tipos de datos, etc.

Por último, en el paso 6 se documenta en la plantilla las condiciones identificadas en el paso anterior para completar la información requerida por la plantilla.
Se observa que si bien es cierto que la planificación y diseño de las pruebas funcionales en la primera etapa de desarrollo del proyecto ayuda a comprobar y verificar que se cumplan los requerimientos dados por el usuario.

Pensando en optimizar el diseño de los casos de pruebas se creó esta metodología que ayude a generar los casos de prueba y que se valide que se cumpla con todas las funcionalidades sin tener mucho grado de dificultad que si sucede con las metodologías descritas anteriormente en los cuales se basó pero que demandarían invertir más tiempo en las pruebas funcionales.

Como trabajo futuro, se plantea la construcción de una herramienta que facilite el uso de la metodología que ha propuesto, así se optimizaría mucho mejor el tiempo a invertir en las pruebas, así como también los recursos que se utilizarán para realizarlos.

3.1.3. Proceso de Desarrollo enfocado en las Pruebas del Software [6]

El proceso fundamental del desarrollo de software contiene una parte importante que son las pruebas del software que se está desarrollando por tanto es importante saber en qué fase o etapa del proceso se debe considerar.

Si bien es cierto las pruebas de software son desarrolladas por personas que pueden cometer errores, por eso es necesario y vital para el proceso probar y probar el aplicativo para minimizar los errores que puedan producir cuando el aplicativo este en producción.

Se comienza desde la etapa inicial del aplicativo, es decir, desde que se tiene la idea o cuando el usuario tiene un requerimiento, se le plantea la idea de cómo se desarrollará el software y todo empieza cuando se acepta la oferta. Dentro de todo el desarrollo del software debe estar contemplado la etapa de las pruebas de software es imprescindible para entregar un producto de calidad.

Para lograr una buena aplicación de las pruebas al software se disponen de diversos estándares que agrupan las mejores prácticas, y a su vez se tienen modelos de evaluación del proceso de las pruebas.

Se describe uno a uno las fases del desarrollo desde el inicio del proceso que comienza cuando el usuario acepta la oferta para iniciar el desarrollo del software. Es así que se consideró importante definir uno a uno al detalle los procesos que se siguen para el desarrollo de las pruebas para así lograr la satisfacción del usuario al entregar la versión definitiva del software y no luego de que ya se hizo la entrega final es decir cuando ya está en producción ya que puede ocasionar el descontento del usuario al encontrarse defectos en la entrega final, por eso es mejor aplicar las pruebas del software y las estrategias de las mismas durante la etapa de desarrollo del software.

Se presenta una descripción detallada del proceso de desarrollo enfocado principalmente en las pruebas del software. Se empieza por describir la parte inicial del proceso que según definición es cuando el cliente acepta que se empiece con el desarrollo del software para

esta parte se debe tener en consideración desde ya las pruebas de software que se aplicarán durante el proceso.

Para ello se basarán en casos anteriores que hayan pasado, por ejemplo, un caso importante es que la mayoría de los defectos de centran en las fases tempranas del proceso de software y el costo de detectarlo aumenta a medida que no se den cuenta de ellos.

Se disponen de diversas metodologías y estándares que apoyarán a las pruebas de software esto en cuanto a la parte inicial. En la parte de Planificación se definirán las técnicas de pruebas a usar, si se apoyará en alguna herramienta, criterios para los casos de prueba, todo esto se debe definir en el Plan de Pruebas. Para el cual se define el alcance de la aplicación, la plataforma donde se probará, quiénes realizarán las pruebas, etc.

Hay consideraciones importantes en esta etapa como que las pruebas deben estar presentes a lo largo de todo el cicla de vida del desarrollo según el modelo en V como se muestra en la Figura 6.

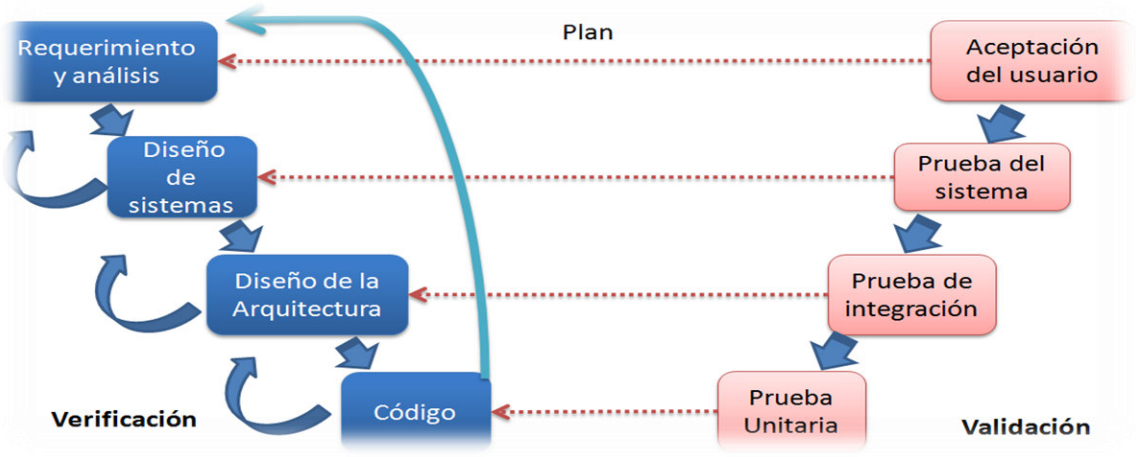


Figura 6: Modelo en "V" Ciclo de Vida del Software enfocado a las Pruebas [WEB05]

A continuación, siguiendo con el ciclo de vida pasamos a la etapa de especificación de los Requisitos donde se verifica que cada característica pueda ser validada es decir probada.

La revisión de los requerimientos nos permitirá identificar y detectar errores en las fases tempranas evitando que nos genere un costo extra es que se detectará al final en la entrega del software.
Continuamos con la etapa de Arquitectura y Diseño, dentro de esta fase tenemos la construcción del aplicativo en donde se realiza las revisiones de código, así como las pruebas unitarias.

Esto nos permitirá obtener un código de alta calidad y que cumpla con los estándares, para esto se puede apoyar de herramientas de análisis de código.
Llegamos a la fase más importante que es la fase de pruebas en donde se tienen diferentes niveles y tipos de pruebas que se pueden aplicar al software.

Tenemos las pruebas de integración, es donde se verifica la funcionalidad de las interfaces entre las distintas partes que componen un sistema. Tenemos también las pruebas de Validación, son aquellas que se realizan sobre un software integrado para evaluar el cumplimiento de los requerimientos.

Las pruebas de Sistema que evalúan el sistema ya integrado teniendo en consideración el rendimiento, la resistencia, la seguridad y la usabilidad. Por último, tenemos las pruebas de Aceptación las cuales se realizan con el usuario en donde se determina si el sistema desarrollado cumple con lo requerido y se obtiene la conformidad del cliente.

Como conclusión se determinó que al desarrollar y explicar uno a uno el proceso de desarrollo apoyado con las pruebas se ha ahorrado tiempo y se logrará que el cliente esté satisfecho con el producto. Se le brinda al cliente la versión definitiva del software desarrollado tras las pruebas de aceptación final con el cliente así ya no saldrán defectos que provocaría un descontento de parte del usuario.

Aun por el contrario faltaría ahondar en la demostración con los datos, los beneficios obtenidos y calcular si sirvió la inversión realizada, se pregunta cómo se obtendría esta información, para este punto se utilizan las métricas que se desarrollará futuramente.

3.1.4. Pruebas de Software en el Ciclo de Vida del Software [7]

Se hace mención el punto de vista de otro autor llamado "James Wittaker" quien brinda algunas ideas de porqué el proceso de probar el software actual identifica varios enfoques que la persona encargada de realizar las pruebas de software debe ser capaz de identificar. La persona encargada de testear el software tiene a su disposición un amplio conjunto de técnicas de prueba, entiende como se utilizará el producto así mismo puede detectar errores comunes que sucedan cuando el software este en producción. Hay que darles más importancia a las pruebas en el contexto del desarrollo en el ciclo de vida del software, es necesario comprender el papel que tienen los probadores y los desarrolladores.

Al ver la situación de las empresas que tienen que buscar a testers que tengan bastante conocimiento de las pruebas que realizan al software para que cuando se haga entrega de este al cliente, éste no encuentre errores de funcionamiento durante su uso.

Para darle solución a esta situación en primer lugar se debe tener más en cuenta las pruebas en la etapa de desarrollo del software y en segundo lugar es necesario comprender la importancia que tienen los testers y los desarrolladores en este proceso.

Es por eso por lo que en este artículo se describen los posibles problemas que puedan encontrar los testers al momento de realizar el proceso de las pruebas, así como se describen 4 fases donde se agruparán los problemas relacionados y que se deben de resolver antes de pasar a la siguiente fase.

Se definieron 4 fases que le permiten a los testers tener una estructura donde pueden agrupar los problemas surgidos y relacionados y que se deben resolver antes de seguir avanzando con el proceso.

La FASE 1: MODELAR EL ENTORNO DE SOFTWARE, consiste en simular la interacción entre el aplicativo y su entorno, para ello se debe identificar y simular las interfaces que va a utilizar el aplicativo.

Las interfaces comunes que se utilizan son: las interfaces humanas, incluyen los métodos con los cuales las personas se comunican con el aplicativo, por ejemplo: los clics del ratón, pulsaciones del teclado, etc. Los testers deciden cómo manejar los datos para comprender cómo se integrarán y así realizar una prueba efectiva.

Las interfaces de software, las cuales indican como utiliza el software al sistema operativo, la base de datos o la librería, en tiempo de ejecución. Asimismo, tenemos las interfaces del sistema de archivos, sólo si son archivos externos.

La FASE 2: SELECCIONAR ESCENARIOS DE PRUEBA, los casos de prueba cuestan tiempo y dinero, el conjunto de casos de prueba que muchos testers eligen son los que cumplan con las metas de cobertura, por ejemplo, cobertura de código fuente, declaraciones de código o la cobertura de entradas. En esta fase los testers se interesan más por las secuencias de declaraciones de código que representan un flujo del aplicativo, se busca el conjunto de escenarios que garantizará encontrar la mayoría de los errores. Así mismo los escenarios de pruebas pueden asegurar que el software funciona de acuerdo con los requerimientos especificados.

La FASE 3: EJECUTAR Y EVALUAR LOS ESCENARIOS, los testers convierten los escenarios en formatos ejecutables, de modo que resulten similares la acción que realizaría un cliente típico, los escenarios de prueba se ejecutan manualmente, debido a esto los testers están buscando la forma de automatizar el proceso, se requiere la simulación de cada dato de entrada y de la salida de todo el entorno. La evaluación implica la comparación de las salidas obtenidas durante la ejecución de los escenarios de prueba con respecto a la salida real del software.

La última FASE 4: MEDIR EL PROGRESO DE LAS PRUEBAS, consiste en contar las cosas, el porcentaje de todo el código revisado, el número de veces que se ha utilizado el aplicativo, el número de veces que se obtuvo respuesta con éxitos, etc.

Los testers deben determinar cuándo parar las pruebas o cuando está listo el aplicativo para entregarlo al cliente. Se necesita medidas cuantitativas de los errores encontrados para disminuir la probabilidad de que el usuario los encuentre cuando este en producción.

Un punto importante para rescatar es que hay que reconocer la naturaleza compleja de las pruebas de software y tomarlas muy en serio, para hacerlo las empresas deben contratar a personas muy capaces y con muchos conocimientos sobre las técnicas y herramientas que se puedan utilizar para realizarlas y asegurar un buen producto de software a entregar al cliente.

Las técnicas y/o fases propuestas en este artículo no deben considerarse como definitivas porque el mundo de las pruebas de software está en constante investigación y desarrollo.

3.1.5. Representación Gráfica de las Pruebas Ágiles [8]

El desarrollo ágil se presenta como alternativa de solución para generar software de calidad, sin embargo, hay varias limitaciones como la mala comunicación dentro del equipo de desarrollo y la carencia de la automatización de las pruebas en las fases del ciclo de vida del software.
Los esquemas preconceptuales son herramientas para representar el conocimiento en cualquier dominio ya que se basa en la lógica preposicional es decir se basa en sentencias lógicas para graficarla, lo cual la convierte en una alternativa viable para el testing ágil, contienen equivalencia semántica y condicionales donde se pueden construir relaciones. En la Figura 7 se presenta una propuesta la cual muestra una aproximación gráfica del proceso del testing ágil. Muestra los pasos a seguir durante el desarrollo del testing, las

tareas y actividades para realizar donde se obtendrá como producto final el software operativo.

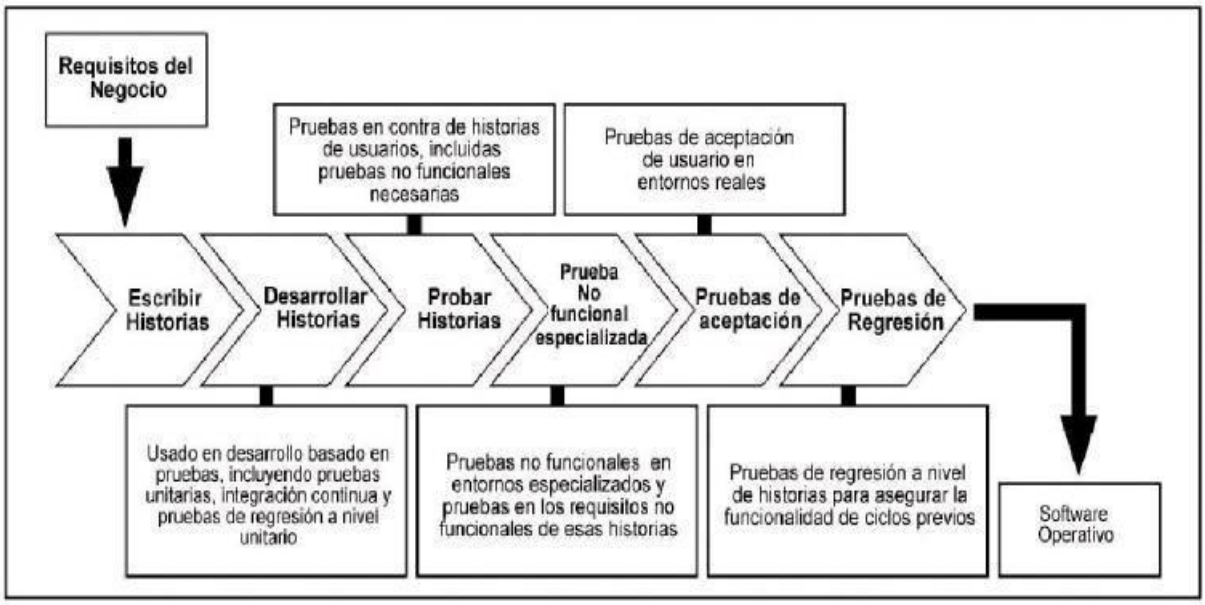


Figura 7. Actividades del testing ágil [7]

El testing ágil es una técnica de pruebas que se está consolidando como propuesta a las metodologías agiles.

En esta metodología todos los miembros pueden ser testers y se realizan actividades de pruebas sobre cada una de las fases del desarrollo ágil de una aplicación de software.

Es por eso que se propone un esquema pre conceptual de testing ágil el cual es un marco de referencia para comprender la técnica utilizada por equipos ágiles en la elaboración de las pruebas de las aplicaciones de software. Las pruebas ágiles surgieron debido a la necesidad de ofrecer un software de calidad y a su vez cumplir con las necesidades del cliente.

Se propone una representación de los fundamentos principales de las pruebas agiles utilizando un esquema pre conceptual como se da a conocer en la Figura 7, donde se pretende extraer la información de manera gráfica. El esquema facilita la comunicación d las ideas de las pruebas ágiles.

Se desarrolló los cuadrantes de las pruebas ágiles, donde se esquematizan gráficamente los tipos de pruebas y el grado de automatización en cuatro cuadrantes, cuyo propósito es ayudar al tester a comprender las técnicas de pruebas y el momento adecuado de cuando probarlas.

Las pruebas en el primer cuadrante (Q1) se orientan a las pruebas de tipo tecnológico que apoyan al equipo, son las pruebas unitarias realizadas por el desarrollador durante el desarrollo del software, y que se puede automatizar utilizando una herramienta en este caso el JUnit.

En el cuadrante (Q2) se agrupan las pruebas orientadas al negocio como, por ejemplo: los prototipos, pruebas funcionales realizadas por pares.

En el Q3 encontramos las pruebas del negocio que critican el producto, para ello se realizan pruebas de exploración, usabilidad y de aceptación.

Por último, en el Q4 se encuentran las pruebas que critican al producto y se orientan a la utilización de una herramienta para verificar los requerimientos no funcionales.

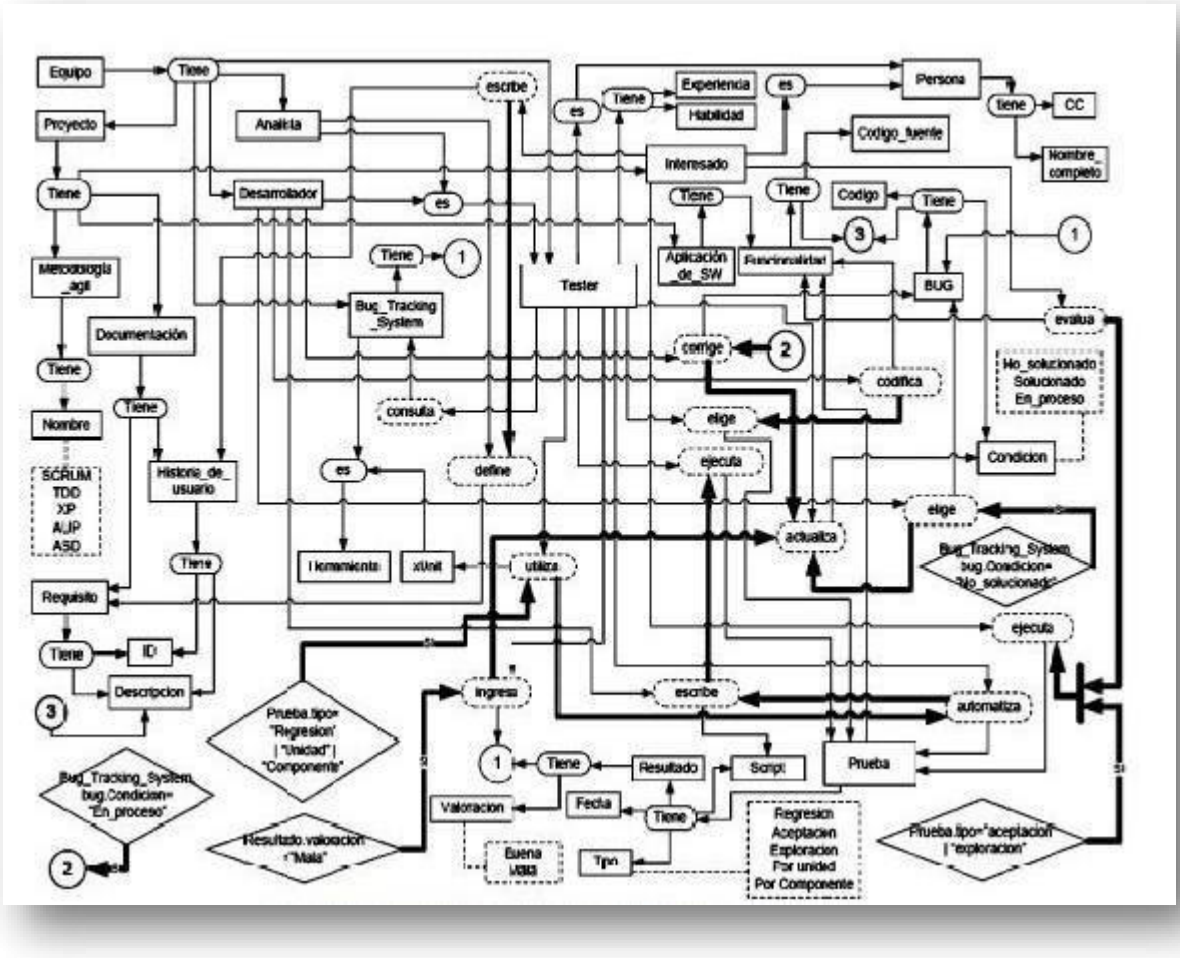


Figura 8. Esquema pre conceptual del testing ágil [6]

Esta representación presentada en este artículo ayuda a mejorar la comprensión del testing ágil, asimismo se usa como inicio para la automatización de esta práctica en las pruebas de software.

Se propone como trabajo futuro la construcción de los tres diagramas básicos de UML a partir del esquema pre conceptual planteado, hacemos referencia al modelo estático mediante el diagrama de clases, y los diagramas de comunicación y máquinas de estados. Además, se pueden generar otros diagramas, producto de la investigación sobre la teoría de los esquemas conceptuales que se vienen realizando.

3.1.6. MANTEMA: a Software Maintenance Methodology Based on the ISO/IEC 12207 Standard [12]

El proceso de mantenimiento de software según algunos autores es la fase más cara en el ciclo de vida de software, dicho esto se denota la importancia de esta fase, pero no es reconocida ni tiene un método en particular por lo cual las organizaciones necesitan una guía completa para ayudar al mantenimiento. Por eso se presenta un ajuste de la norma ISO / IEC 12207 para los procesos del mantenimiento del ciclo de vida.

Esta norma es aplicable a la adquisición de sistemas de software, productos y servicios para el suministro, desarrollo, operación y mantenimiento de productos de software, y para la parte de software de firmware.

En él se describe la arquitectura de los procesos del ciclo de vida, pero no detalla cómo implementar las actividades y tareas que implica este tipo de procesos. Esta norma es para diferentes modelos de ciclo de vida y su adaptación al ciclo de vida utilizado es la responsabilidad de cada parte. [12]

La ISO / IEC 12207 describe 5 procesos principales como las actividades de:

Adquisición: Es referido a la organización que obtiene un sistema o producto de software.

Suministro: La organización que proporciona el sistema o producto de software.

Desarrollo: La organización define y desarrolla el producto de software.

Operación: La organización que provee lo necesario para operar un sistema informático en su entorno para sus usuarios.

Mantenimiento: Administra modificaciones al producto de software para mantenerlo actualizado y en forma operativa. Este proceso incluye la migración y el retiro del producto de software.

Activity	*Task no.*	*Task*		
Process implementation	1	Developing of maintenance plans		
	2	Establishing procedures for receiving and recording problems reports and modification requests		
	3	Implementing the Configuration Management Process (an organizational process)		
Process and modification analysis	4	Problem report or modification request analysis		
	5	Problem verification or replication		
	6	Developing options for implementing the modification		
	7	Problem documentation		
	8	Obtaining aprobación para la modificación seleccionada option		
Modification implementation	9	Conducting analysis and determining which documentation, software units and versions needing to be modified		
	10	Entering at the Development Process to implement the modifications	10.1	Process implementation
			10.2	System requirement analysis
			10.3	System architectural design
			10.4	Software requirement analysis
			10.5	Software architectural design
			10.6	Software detailed design
			10.7	Software coding and testing
			10.8	Software integration
			10.9	Software qualification testing
			10.10	System integration
			10.11	System qualification testing
			10.12	Software installation
			10.13	Software Versión de aceptación support
Maintenance review/Acceptance	11	Conducting reviews, with the organization autoriza la modificación, para determinar la integrity of the modified system		
	12	Obtaining approval		
Migration	13	Accordance with ISO/IEC 12207		
	14	Developing a migration plan		
	15	Notifying users of the migration plans and activities		
	16	Parallel operations of the old and new environments		
	17	Notifying users of the arrival of the migration		
	18	Post-operations review		
	19	Saving old environment data		
Retirement	20	Retirement plan		
	21	Notifying users of the retirement plan and activities		
	22	Executing parallel operations		
	23	Notifying users of the retirement		
	24	Saving old environment data		

Figura 9. Activities and tasks in the ISO/IEC 12207 Processor de Mantenimiento [12]

El aporte del artículo es que proponen una metodología tal cual se ve en la Figura 9, la cual contiene una combinación de actividades y tareas que varían depende del tipo de

mantenimiento que se está realizando. Se integran algunos procesos como verificación, validación, auditoría, resolución de problemas y gestión además en la metodología se presentan los procesos como se muestra en la Figura 10.

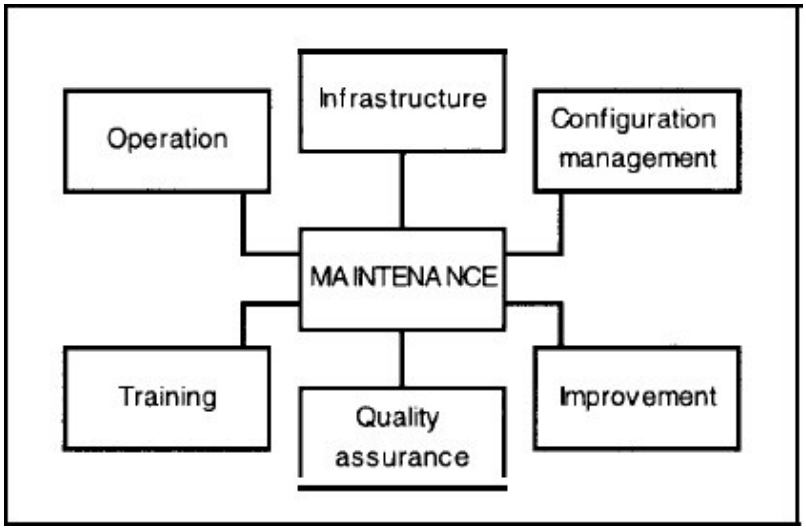


Figura 10: Procesos de la Metodología Propuesta MANTEMA

Concluimos que la metodología y los nuevos trabajos se encaminan a la aplicación de métricas para el mantenimiento del software, es decir, poniendo especial atención a las métricas para grandes sistemas y nuevos entornos.

3.2. Herramientas de Apoyo

3.2.1. Automatización de las Pruebas Funcionales con Herramientas Open Source [9]

Se define una metodología para la automatización de las pruebas funcionales adicionando un conjunto de herramientas open source para facilitar el desarrollo de estas.

Para realizar la automatización de las pruebas hay diversas herramientas de apoyo así tenemos: las pruebas estáticas, herramientas para la revisión del análisis y el modelado; herramientas para el diseño de los casos de pruebas; herramientas para la ejecución de los casos de prueba y por último las herramientas para la carga y monitoreo.

Actualmente en el Centro de Ensayos de Software (CES) utilizan un proceso llamado ProTest el cual es un proceso de testing funcional independiente, pero en el presente

artículo se describe una extensión del proceso, así como nuevas actividades específicas para la automatización.
Se apoyó en las herramientas del grupo Selenium (Selenium Core, IDE y Remote Control) para la ejecución de las pruebas, estas permiten crear y ejecutar pruebas automatizadas sobre aplicativos webs.
Se analizó la situación de la entidad llamada "Centro de Ensayos de Software" (CES) la cual agrupa la mayoría de las empresas productoras de software de Uruguay las cuales ofrecen diversos servicios tales como: servicios de prueba independiente, consultoría y capacitación.

Algunos de los clientes del CES, son pequeñas o medianas empresas que demostraban su interés en automatizar las pruebas funcionales que realizaban, ya que dichas empresas comercializan pocos productos de software, los cuales están en constante mantenimiento y mejora. Por tanto, necesitan asegurarse de que cuando se realicen los mantenimientos y/o mejoras no afecte el funcionamiento principal del producto.

Por esto es importante automatizar las pruebas y así tener la seguridad que no surgirán defectos cuando se haya puesto en producción así mismo se analizó usar herramientas opensource para reducir los costos y el tiempo que se necesiten para dichas pruebas.

Se define la metodología utilizada tiene una serie de pasos y /o actividades definidas en la Figura 11. Comenzamos con la actividad AT1 la cual trata de la definición de las pruebas automatizadas, actividad donde se define las funcionalidades que con la ayuda de las herramientas automatizadas se ejecutarán las pruebas.

En la AT2 tenemos la definición de los procedimientos de prueba, en la cual se definen los casos de pruebas y los scripts que conformarán las pruebas automatizadas. Se debe especificar cada paso que se realizará y las verificaciones que se realizarán durante las pruebas.

Continuamos con la AT3, generación de casos de pruebas y los scripts, se obtiene los casos de pruebas con sus scripts correspondientes así mismo se verifica la correcta funcionalidad del script ejecutado.

En el AT4, ejecución de las pruebas automatizadas del ciclo funcional, el objetivo principal es ejecutar una prueba completa de los casos de pruebas correspondientes al ciclo funcional y verificar que se ejecute correctamente en el entorno de pruebas preparado para dicho fin. En la metodología propuesta se utilizará herramientas opensource, las cuales se necesitan investigar, eso se da en el paso AT5, donde se investiga y/o se modifican las herramientas que se están utilizando para la automatización.

En la actividad AT6 se configura el entorno en el cual se van a desarrollar las pruebas funcionales automatizadas. Como penúltimo paso AT7 tenemos la validación de las pruebas automatizadas en el entorno del usuario, se realizan las pruebas y se verifican que los scripts y los casos de pruebas están funcionando correctamente. Por último, en la actividad AT8 se organizan y gestionan los artefactos obtenidos de las pruebas funcionales automatizadas.

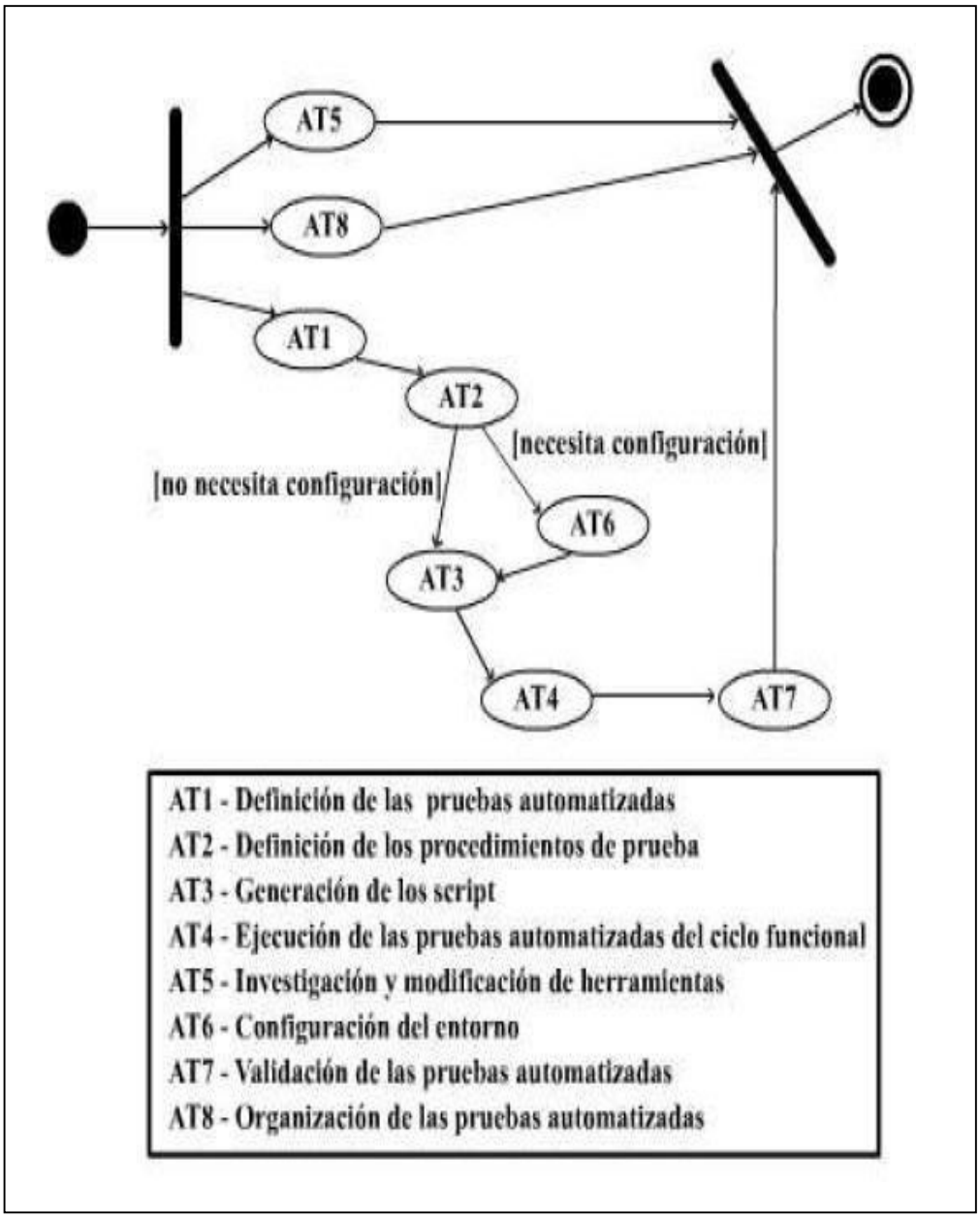


Figura 11. Diagrama de Actividad de la Metodología Propuesta. [9]

En la figura 12 se presentan las actividades descritas líneas arriba, los roles y artefactos que se tendrán en la metodología propuesta. Los roles pueden ser: Líder, Diseñador de pruebas, el usuario y /o cliente y el tester. Se muestran los documentos de entrada necesarios para comenzar con las pruebas funcionales automatizadas ya que contienen la información esencial y necesaria para poder realizar correctamente los casos de pruebas. Así mismo tenemos los documentos de salida los cuales se generan luego de realizada las pruebas funcionales como por ejemplo la evidencia de las pruebas, los scripts ejecutados, las herramientas utilizadas y toda la documentación pertinente.

ACTIVIDAD	ROLES	ENTRADA	SALIDA
AT1 - Definición de las pruebas automatizadas	Líder Diseñador de Pruebas Cliente	Requerimientos Acta reunión cliente	Documento que especifica las funcionalidades y ciclos funcionales a probar con las pruebas automatizadas
AT2 - Definición de los procedimientos de prueba	Diseñador de Pruebas Cliente	Requerimientos Acta reunión cliente Documento obtenido en AT1	Documento con Suites y Scripts que componen las pruebas
AT3 - Generación de suites y scripts	Tester	Documento obtenido en AT2 y documento obtenido en AT6	Suites y Scripts que las componen
AT4 - Ejecución de las pruebas automatizadas del ciclo funcional	Tester	Suites y Scripts obtenidos en AT3	Suites y Scripts verificados
AT5 - Investigación y modificación de herramientas	Tester	Herramienta a investigar	Nuevas herramientas (adquiridas o modificadas) y documentación pertinente
AT6 - Configuración del entorno	Tester	Aplicaciones a probar y documento obtenido en AT2	Aplicaciones aptas para ser probadas y documentación pertinente
AT7 - Validación de las pruebas automatizadas	Cliente Tester	Suites y Scripts obtenidos en AT4	Suites y Scripts verificados y validados
AT8 - Organización de las pruebas automatizadas	Diseñador de Pruebas Tester	Artefactos generados en el proyecto	Artefactos gestionados

Figura 12. Actividades, Roles y Artefactos de Entrada y de Salida de la Metodología Propuesta. [9]

Después de desarrollar la metodología se observó que se necesitaba de alguna herramienta que permita gestionar los artefactos que se generan durante la automatización de las pruebas funcionales, luego de investigar se llegó a la conclusión que se utilizaría la herramienta Fitnesse y Selenium Remote Control para crear y gestionar los artefactos de manera adecuada, ya que el objetivo principal es crear una interfaz sencilla que permita ejecutar comandos similares a Selenium. Se propone integrar las dos herramientas para mejorar la organización de las pruebas de manera colaborativa con el equipo de pruebas.

CAPÍTULO IV: APORTE TEÓRICO

Se creará un nuevo proceso basado en el modelo que plantea la Norma Peruana NTP-ISO/IEC 12207:2016, se describirá paso a paso las actividades a realizar durante el proceso de desarrollo de las pruebas funcionales, en donde se utilizará la herramienta Selenium para aprovechar las funcionales que éste contiene y así automatizar el proceso de las pruebas funcionales.

Aplicando lo establecido por la norma técnica peruana en sus capítulos de Verificación y Validación, la tesis justifica un esfuerzo de aplicar la validación por eso se establece un proceso para validar el producto de software. Se seleccionan las tareas de validación que se definen a continuación, incluyendo los métodos, técnicas y herramientas asociados para realizar las tareas. Para ellos se cuenta con la herramienta de apoyo para la automatización de las pruebas funcionales seleccionado de acuerdo con las funcionalidades que tiene el aplicativo.

El plan de validación a utilizar contendrá los siguientes puntos:

- Cantidad de casos de pruebas fallados y pasados después de la ejecución de los casos.
- Métricas obtenidas del comportamiento después de la ejecución.
- Reportes de tiempo de ejecución del proceso ejecutado.

También se debe implementar un plan de verificación. Los problemas y no conformidades detectadas en el trabajo de verificación se deben documentar y se deben resolver. Luego los resultados de las actividades de verificación son puestos a disposición del cliente y de las otras partes interesadas.

El plan de verificación contiene las siguientes características según la norma, la cual menciona:

:

“Verificación de los requisitos y/o requerimientos: Los requisitos se deben verificar considerando los criterios que se enumeran a continuación “Los requisitos del sistema son consistentes, factibles y se pueden probar.

- Los requisitos del sistema se han asignado adecuadamente a los elementos de hardware, elementos de software y operaciones manuales de acuerdo con los criterios de diseño.

Verificación del diseño de los casos de pruebas:

- El diseño es correcto, consistente y con trazabilidad hacia los requisitos.
- El diseño implementa la secuencia correcta de eventos, entradas, salidas, interfaces, flujo lógico, asignación de tiempo y presupuestos a medida, así como definición, aislamiento y recuperación de errores.
- El diseño seleccionado se puede derivar de los requisitos.

Verificación del código:

- El código es trazable hasta el diseño y requisitos, comprobable, correcto y cumple con los requisitos y los estándares de codificación.
- El código implementa la secuencia adecuada de eventos, las interfaces consistentes, datos correctos y flujo de control, completitud, asignación adecuada de tiempo y presupuesto a medida, así como definición, aislamiento y recuperación de errores.

Verificación de la documentación generada por el proceso:

- La documentación es adecuada, completa y consistente.
- La preparación de la documentación es oportuna”. [11]

4.1 Descripción del Proceso Propuesto:

En el proceso creado tenemos las siguientes actividades definidas basadas en las actividades descritas en la norma peruana NTP 12207:2016 y los cuales describiremos a continuación:

Definir los Requerimientos:

Para la definición de los requisitos de acuerdo con la norma se establece que los requisitos deben ser fiables y que se puedan probar por eso el primer paso es la definición de los requerimientos, esta actividad se debe desarrollar con el usuario ya que esta etapa es primordial para pasar a la siguiente fase que es el análisis. Asimismo, en esta actividad se debe contar con el entendimiento claro del flujo del negocio, lo cual es necesario, ya que nos permitirá determinar los posibles escenarios donde se realizarán los casos de prueba y así poder realizar la mayor cantidad de pruebas funcionales.

Para tener una buena definición de requerimientos según lo menciona la página sobre la Metodología Gestión de Requerimientos: “es necesario realizar una buena identificación de estos, posterior a esto es importante definirlos de manera detallada, donde se involucre la información aportada por los usuarios.

Para realizar con éxito la definición de los requerimientos es importante conseguir que los requerimientos sean claramente definidos para minimizar la ambigüedad de los requerimientos, para esto es importante tener en cuenta lo siguiente:

- Definir los requerimientos teniendo en cuenta la información identificada con la perspectiva del usuario.
- Reutilizar requerimientos, revisando proyectos ya finalizados para ver si contienen material potencialmente reutilizable. La ventaja de esta reusabilidad es que, una vez que un requisito ha sido especificado satisfactoriamente para un producto y que el producto ha tenido éxito, el requerimiento no tendrá que volverse a inventar, podrá ser utilizado las veces que se desee teniendo en cuenta la autoría.

- Se debe documentar los requerimientos de una forma clara y correcta. En la mayoría de los proyectos se observa que la documentación de los requerimientos puede parecer una tarea tediosa, pero es la única manera de asegurar que la esencia de los requisitos ha sido capturada correctamente, y que esto pueda ser probado".

  Las características de un requerimiento son sus propiedades principales. Un conjunto de requerimientos en estado de madurez, deben presentar una serie de características tanto individualmente como en grupo. Las características más importantes son:

  - Necesario: Es necesario si el no tenerlo en cuenta provoca una deficiencia en el sistema a construir, y además su capacidad, características físicas o factor de calidad no pueden ser reemplazados por otras capacidades del producto o del proceso porque no funcionaría igual.
  - Conciso: Es conciso si es fácil de leer y entender. La redacción debe ser simple y entendible para aquellos que vayan a necesitarlo en un futuro.
  - Completo: Está completo si no necesita ampliar detalles, es decir, si se proporciona la información necesaria para su comprensión.
  - Consistente: Es consistente si no está en contraste con otro requerimiento.
  - No ambiguo: Cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector.
  - Verificable: Cuando puede ser cuantificado que permita hacer uso de los siguientes métodos de verificación, inspección, demostración o pruebas.

Tener en cuenta que se debe realizar una reunión con el usuario para elaborar la Lista de Requerimientos y definir el alcance del proyecto a realizar. El Analista de Pruebas debe participar para que pueda tener una visión de lo requerido por el usuario y así ir definiendo sus escenarios para los casos de pruebas.

OUTPUT: Documento Lista de Requerimientos. – Ver ANEXO 01

Elaborar los Casos de Pruebas:

ESCENARIO DE CASOS DE USO [10]

Según lo mencionado en el libro de JACOBSON, IVAR; G. BOOCH Y J. RUMBEAUGH sobre los escenarios de casos de uso: "Para la generación de casos de prueba son importantes los escenarios de casos de uso. Un escenario es una instancia de un caso de uso o de manera similar: un flujo completo de eventos a través del caso de uso. Así, el flujo básico de un caso de uso constituye un escenario, y cada flujo alterno que se desprende del flujo básico sería otro. Por tanto, se puede decir que cada caso de uso describe varios escenarios de uso, y estos se pueden dividir en:

- Los escenarios correspondientes al flujo básico del caso de uso.
- Los escenarios que incluyen al menos un flujo alterno del caso de uso.
- Los escenarios que producen al menos una excepción.
- Estos escenarios van a constituir las bases para la creación de los casos de prueba.

Un caso de prueba especifica qué probar en el sistema y está formado, además del nombre y una descripción opcional, por un conjunto de entradas de prueba, de condiciones bajo las que se deben realizar las pruebas, y de resultados esperados".
Un caso de prueba elaborado incluye:

- "Un nombre que identifica el caso.
- Sus entradas: es la secuencia exacta de acciones que debe realizar el usuario para ejecutar el caso de prueba. Note que, si las acciones involucran introducir valores, deben especificarse los valores exactos que deben introducirse.
- Su salida esperada: Los resultados exactos observables que debe producir el sistema al ejecutar el caso de prueba.

Idealmente debemos elaborar, en el menor tiempo posible, un conjunto mínimo de casos de prueba que garantice el nivel de calidad deseado para el software. En la realidad, debemos conformarnos con elaborar, en un tiempo razonable, un

conjunto reducido de casos de prueba, tal que, si el proceso de prueba cumple con el criterio de terminación, la probabilidad de satisfacer el nivel de calidad deseado es alta".

Seguidamente, luego de determinar los posibles escenarios, se procede a elaborar los casos de prueba, con el conocimiento de los requerimientos y aspectos vitales del negocio, se elabora una matriz de casos de pruebas funcionales, las cuales validarán que se hayan cumplido los requerimientos solicitados por el usuario en la etapa de definición de los requerimientos.

Se describe paso a paso el flujo que se debe seguir para la realización de los casos de prueba.

Se llena el documento de acuerdo con el documento de referencia que se anexa en Documento Matriz de Casos de Pruebas – Anexo 02 así como también se puede trabajar con la herramienta Opensource llamada Testlink.

Evaluar los casos de pruebas.

Para ejecutar los casos de pruebas creados, primero el jefe de proyecto debe evaluar los casos y si éstos satisfacen los requerimientos dados por el usuario. Si se llegara a encontrar alguna inconformidad u observación se hace del conocimiento del analista de pruebas para que evalúe y re-diseñe el caso de prueba con las observaciones encontradas.

Si al evaluar los casos de pruebas, el jefe de proyecto da su conformidad, se prepara el entorno, similar al que se tendrá cuando el software pase a producción, para poder iniciar con la ejecución de los casos de prueba.

Enviar observaciones al Analista de Pruebas para reelaboración de los casos.

Luego que el Jefe de Proyecto revisa la matriz de casos de Pruebas, si algún caso no cumple con lo especificado en la lista de requerimientos o no está completamente correcto, el Jefe de Proyecto realiza las observaciones encontradas y luego envía las observaciones al Analista de Pruebas para que corrija los casos y proceda a ejecutar los casos.

Validar los casos y proceder a pedir ejecución de los casos de pruebas.

El Jefe de Proyecto se encarga de volver a validar los casos corregidos y reelaborados, valida los casos descritos verificando que cumplan los requerimientos descritos en la Lista de Requerimientos.

Después de validar los casos, se procede a enviar la aprobación para la ejecución de los casos que quedaron pendientes de validar.

Preparar el entorno para las pruebas.

Entorno de pruebas es el entorno que contiene hardware, instrumentación, simuladores, herramientas software y otros elementos de soporte necesarios para realizar una prueba.

CARACTERÍSTICAS DE UN AMBIENTE DE PRUEBA: Según se menciona en la página de software testing:

- "Debe residir en un computador distinto al personal del desarrollador. Preferiblemente en un servidor compartido por los equipos de pruebas de software.
- Utilizar nombres de dominio (no direcciones IP) distintos a los de producción y desarrollo para evitar confusión del personal de pruebas.
- Ser lo más similarmente posible al ambiente de producción, incluyendo:

  Aplicaciones locales, cliente servidor, web, etc.

Configuración de servidores.

Manejadores de Bases de datos de producción, de las mismas versiones.

Configuración de base de datos.

Cuentas de usuario de sistema operativo, bases de datos y aplicaciones con la misma configuración y privilegios de acceso.

Componentes de infraestructura deben ser similares (Ej. Clusters).

Versiones de software.

Réplicas de todos los componentes con los que el Software desarrollado tendrá interoperación, incluyendo: Otras aplicaciones, otros middlewares, interfaces, demonios (Daemons), utilidades FTP, entre otros.

- Si no es posible tener instalado el mismo manejador de base de datos que en producción y desarrollo, automatizar la propagación de cambios de un ambiente a otro (Existen herramientas de software que lopermiten).

- Instalar las herramientas de gestión de casos de prueba y gestión de incidencias en una máquina o servidor distinto al del ambiente de pruebas integrales, compartido por los equipos de pruebas.

- Capacidad de servir a múltiples audiencias, por ejemplo, administradores de sistemas, usuarios finales, desarrolladores. En todos los casos sólo para ejecución de pruebas.

- Debe apoyarse en herramientas de control de versiones, especialmente cuando existen múltiples proyectos en paralelo probando distintas funcionalidades".

Se tiene en cuenta que luego de haber validado la matriz de casos de pruebas se procede a preparar el ambiente para la ejecución de las pruebas.

Ejecutar los casos de pruebas en el entorno.

Al comenzar a ejecutar los casos de prueba, primero se procede con el subproceso de automatización de las pruebas funcionales el cual se apoyará con la herramienta Selenium, se escogió de entre las diferentes herramientas para pruebas que existen, ya que contiene diversas funcionales funcionalidades necesarias para una óptima realización de las pruebas funcionales al aplicativo.

Selenium tiene la ventaja de ser una herramienta de código abierto. Esta herramienta está diseñada para navegadores, lo cual es conveniente ya que el aplicativo a probar está desarrollado en java web. Se puede escribir los scripts tanto en C#, Java, Groovy, Perl, PHP, Python y Ruby.

La herramienta Selenium tiene como principal objetivo a la automatización de pruebas sobre aplicaciones web, su uso no se limita a esta actividad, ya que aquellas tareas repetitivas a través del navegador pueden y deberían también automatizarse.

El potencial de esta herramienta es que puede ser utilizado para la grabación de las pruebas funcionales durante la Generación de pruebas. Con esto se consigue obtener un conjunto de pruebas automatizadas que podrán ser utilizadas cuando sea necesario para repetir las pruebas, es decir guarda un histórico de las pruebas para una reutilización de ser necesaria.
Otra ventaja que tiene Selenium según Toni Cárdenas:" es que consta de varias herramientas. Por un lado, Selenium IDE es una extensión para Firefox que registra nuestra actividad en el navegador durante un período determinado; esta actividad se traduce en una serie de comandos que podremos repetir y repetir. Esta serie de comandos puede exportarse en forma de script en muy diversos lenguajes: HTML, Python, Ruby, Java (JUnit), C# y algunos más.

Este script generado puede editarse para posteriormente ser ejecutado por Selenium WebDriver. Éste implementa una API cliente lista para ser usada con tu entorno de testing escogido, que se ejecutará sobre un servidor que gestiona los principales navegadores que tengas instalados para realizar en ellos las pruebas. Es decir, podrás ejecutar las pruebas automáticamente en todos los navegadores relevantes, de forma que no se escapa ningún detalle del aplicativo".

Registrar errores encontrados.

Al terminar de ejecutar los casos de pruebas, se presentan dos casos, los casos que se prueban con éxito y se llega al resultado esperado, y los casos que no se logra obtener la respuesta positiva además de no obtener el resultado esperado, se procede a reportarlo y a evidenciarlo para que se corrija, se hace del conocimiento al programador para que subsane los errores encontrados.

Asimismo, para el proceso de verificación que se contempla también en el proceso elaborado, los defectos se identifican y se registran para el posterior levantamiento de los mismos.

Se procede a elaborar una matriz de observaciones para colocar los pasos realizados además de identificar y mapear el error encontrado en el sistema.

Ver Anexo 03- Matriz de observaciones.

Guardar histórico de los casos de éxito.

Se describen paso a paso cada una de las actividades realizadas en cada caso de prueba que se ejecutó correctamente y pasó la prueba de validación cumpliendo los requerimientos descritos inicialmente en la Lista de Requerimientos, es importante guardar cada caso realizado con éxito para poder reutilizarlo si es que fuera necesario en algún otro aplicativo o se desea revisar más adelante los paso que se realizaron durante los casos de prueba.

Para guardar los casos se usará la herramienta Testlink, la cual es una herramienta gratuita según la página de Gustavo Terrena la cual menciona: "que el Testlink

permite crear y gestionar casos de pruebas y organizarlos en planes de prueba. Estos planes permiten a los miembros del equipo ejecutar casos de prueba y registrar los resultados dinámicamente, generar informes, mantener la trazabilidad con los requerimientos, así como priorizar y asignar tareas.

Esta herramienta ofrece:

- Soporte para diferentes productos.
- Creación de diferentes roles con distintos permisos para los integrantes del equipo de testing.
- Creación ilimitada de carpetas en forma de árbol (llamadas test - suites) para una mejor organización y agrupamiento de los casos de prueba.
- Versionado de casos de prueba.
- Ejecución de los casos de prueba por versión del software bajo prueba.
- Creación de plan de pruebas para la ejecución.
- Asignación de casos (para ejecución) al usuario. De modo que en una misma prueba podemos tener varios testers trabajando y ellos sólo ven sus casos de prueba asignados.
- Soporte para relacionar los casos con requerimientos/especificaciones.
- Generación de distintos tipos de reportes: generales del producto, por plan de pruebas, gráficos, etc.
- Posibilidad de mandar los reportes por mail al probador".

Características

- Roles de usuario y gestión.
- Agrupación de casos de prueba en las especificaciones de prueba.
- Planes de prueba.
- Plataformas.
- Con los requisitos de control de versiones y revisiones.
- Apoyo para probar diferentes construcciones de software.
- Informes, gráficos y monitores.
- Integración con otro software utilizando una API proporcionada.

- La integración del sistema de seguimiento de fallos (Mantis, JIRA, Bugzilla, FogBugz, Redmine, y otros).

Reportar errores encontrados en el aplicativo.

Al realizar la ejecución de los casos de pruebas descritos en la matriz de los casos de pruebas y obtener los casos fallados, se procede a reportar las fallas al Jefe de Proyecto para que éste le asigne al responsable, quien se encargará de levantar las observaciones.

Reporte de errores para análisis y levantamiento.

Luego de encontrar los casos fallados reportados, se recibe dicho reporte y se analiza paso a paso el detalle del caso donde falló la ejecución del caso de prueba para levantar la observación encontrada.
Cada uno de los casos fallados se encontrará en la matriz de observaciones con una pequeña descripción de los pasos seguidos y la evidencia de lorealizado.

Corregir los errores encontrados.

Después del análisis se procede a corregir una a una las observaciones encontradas y reportadas en la matriz de observaciones del proyecto.

Se corrige el error y se verifica el flujo a continuar durante cada paso a realizar de los casos de prueba.

Luego de corregirlos, se procede a enviar los cambios a Analista de Pruebas para que vuelva a ejecutar los casos fallados obteniendo una respuesta positiva.

Volver a ejecutar casos de errores.

Luego de que el programador solucione las observaciones encontradas, el Analista de pruebas vuelve a ejecutar los casos de prueba que no tuvieron éxito.
Al conocer el flujo se le hace más práctico volver a ejecutar uno a uno los casos fallados que generaron error en la primera ejecución.

Realizar pruebas con el usuario.

Al finalizar, la siguiente actividad es realizar las pruebas con el usuario en su ambiente para luego firmar con él, el Plan de Aceptación de pruebas, donde se da constancia de que los involucrados están conformes con las pruebas realizadas y que se cumplen los requerimientos solicitados.

Se solicita una reunión con el usuario para realizar las pruebas del sistema, así como se les hace una pequeña inducción de los pasos a seguir durante el desarrollo de cada caso de prueba descrito en la matriz de observaciones.

Validar que las pruebas validan el requerimiento.

Al terminar de validar todos los casos de la matriz de observaciones se procede a revisar si cada uno de los casos descritos cumple con los requerimientos descritos en la Lista de Requerimientos.
Se verifica que cada caso maneja su propio flujo de negocio.

Firmar acta de aceptación de pruebas.

Al terminar de ejecutar los casos y obtener un resultado positivo de la suite probada, se procede a firmar el acta de aceptación de las pruebas realizadas donde se establece la inducción que se le hizo a cliente. Asimismo, se le pone en conocimiento todo el flujo del negocio aplicado al sistema y se le realiza una pequeña inducción de las funcionalidades nuevas agregadas para el pase a producción.

Para ver el acta de capacitación ver Documento de Acta de Aceptación – Anexo 04.

Firmar Acta de aceptación y registrarlo.

El fin principal es obtener el Plan de Aceptación de pruebas ya que este documento valida el correcto despliegue de los casos de prueba y la satisfacción del usuario con el aplicativo desarrollado y presentado.

Concluir el proceso de las pruebas funcionales.

Finalmente se cierra el proceso de las pruebas funcionales al entregar toda la documentación relacionada al pase producción que se está ejecutando debidamente validada. Se cierra la fase de Pruebas funcionales del aplicativo.

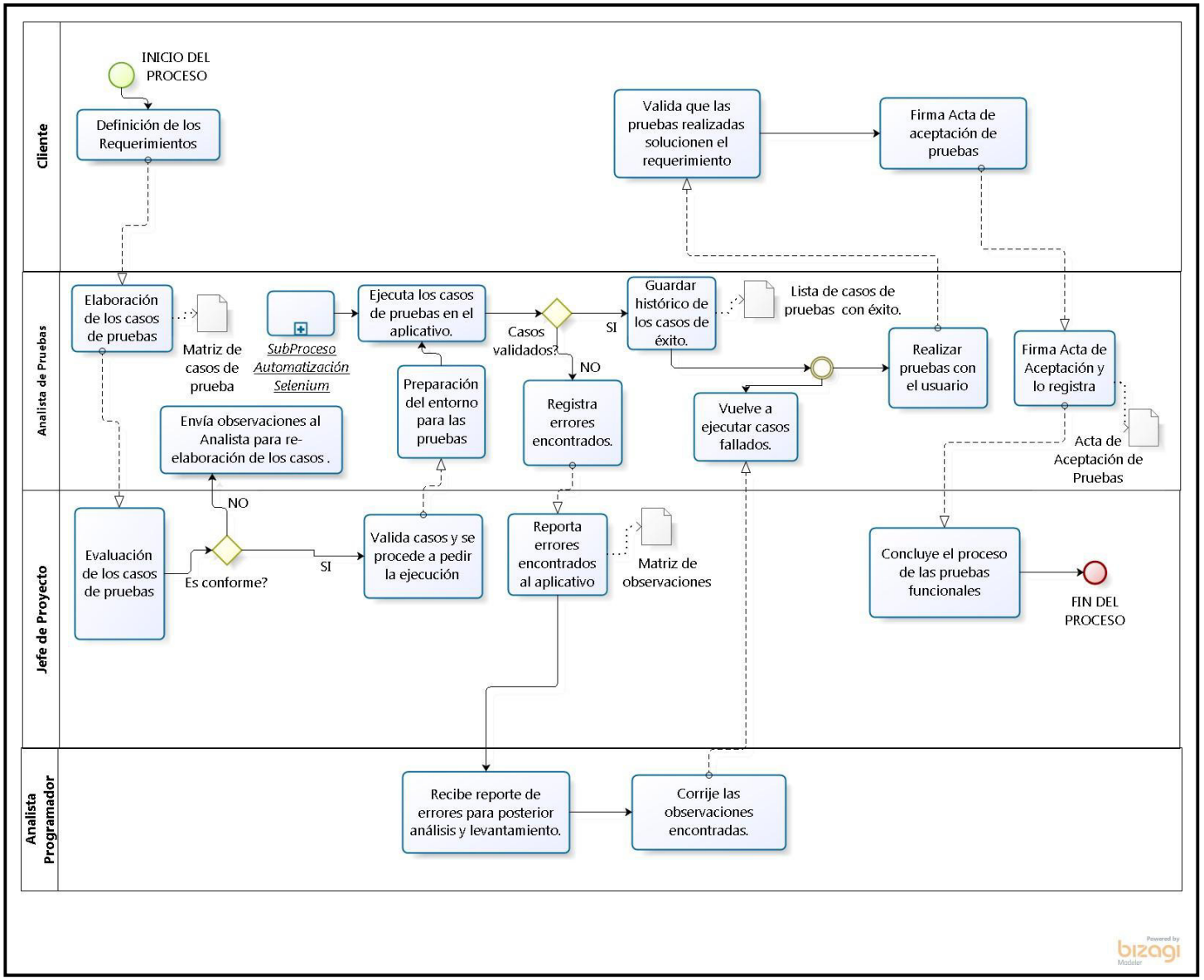


Figura 13. Proceso de Pruebas Funcionales Propuesto. [Elaboración Propia]

Proceso de Automatización con la Herramienta Selenium

Este proceso de Automatización de Pruebas Funcionales sirve de apoyo al modelo para la Automatización de las Pruebas Funcionales.

Se siguen los siguientes pasos para el proceso:

Seleccionar del aplicativo: El primer paso a seguir es identificar el aplicativo en el cual se realizarán las pruebas funcionales para ello, se debe tener en consideración que debes escoger un aplicativo web en java.
Evaluar e identificar los flujos que tiene el aplicativo.

Inicializar de las pruebas funcionales: Paso a paso se sigue el flujo por cada una de las funciones que tiene el aplicativo web validando contra la matriz de casos de pruebas. Se validan uno a uno cada caso de prueba siguiendo los pasos descritos en la misma.

Ejecutar de los casos de pruebas: Verificar la correcta funcionalidad de cada uno de los casos de pruebas declarados en la matriz y obtener el resultado esperado por cada uno.

Verificar la grabación de los casos de prueba: Siempre validar que la herramienta esté grabando paso a paso el flujo que se está siguiendo para el caso ejecutado.

Repetir el flujo de los casos de pruebas: Al terminar un caso de prueba determinado se procede a repetir el flujo en automático para validar que se guardaron correctamente los pasos descritos en la matriz y que el resultado esperado es el correcto.

Reportar los errores de los casos de pruebas: En el caso de surgir errores durante la ejecución de los casos de pruebas, se procede a listar en una matriz de observaciones del aplicativo adjuntando la evidencia del flujo ejecutado para posterior evaluación y levantamiento para proceder a volver a validar.

Evidenciar el flujo continuado: Al terminar todos los flujos de la matriz de casos de pruebas, se procede a guardar como evidencia los casos ejecutados sean de éxito o de falla. Así se guarda en automático las pruebas realizadas para volver a ejecutarlas automáticamente y optimizar el tiempo invertido en las pruebas funcionales.

Recepción de las correcciones de los casos de pruebas: Finalmente al entregar la matriz de observaciones encontradas durante la ejecución de los casos de pruebas se le hace entrega al Jefe de Proyecto para que levante las observaciones y se puedan volver a validar siguiendo de nuevo el proceso.

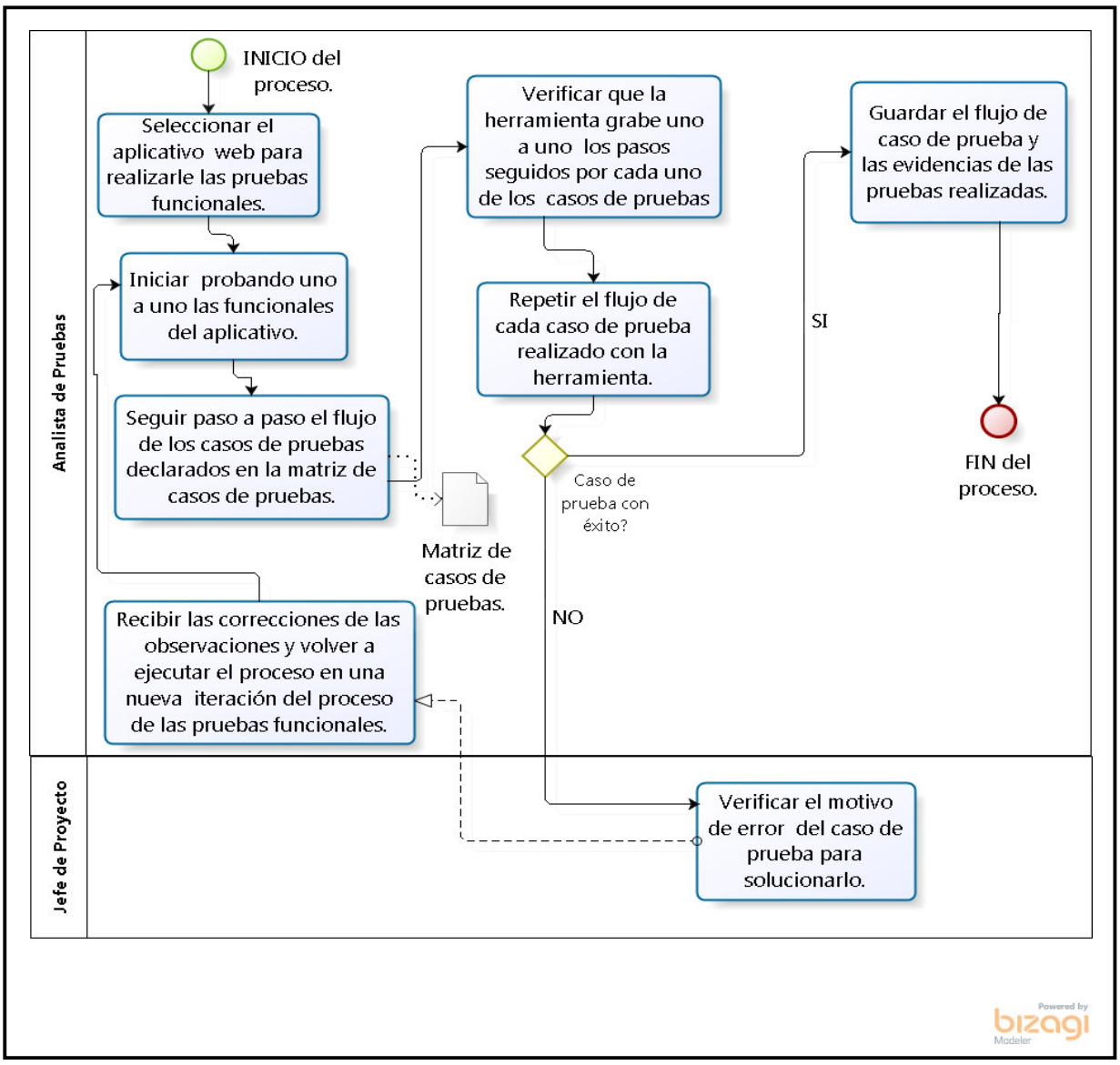


Figura 14. Proceso de Automatización de Pruebas Funcionales usando la herramienta Selenium IDE [Elaboración Propia]

4.2. Comparación de Herramientas para Automatización [WEB04].

En el cuadro mostrado se denotan la comparación de las características que contienen diversas herramientas para la automatización, esto para escoger la que mejor nos ayudará en el proceso de la realización de las pruebas funcionales de acuerdo con las características del aplicativo que se va a probar.

Las herramientas para comparar son las siguientes según la definición mencionada en la página de Globe ***[WEB04]:***

"UNIFIED FUNCTIONAL TESTING: Unified Functional Testing era antes conocida como Quick Test Professional, QTP. Soporta una variedad muy extensa de tecnologías: Java, SAP, Siebel, Visual Basic .Net y Oracle entre muchas otras.

Esta Herramienta se basa en el reconocimiento de objetos, aunque se puede utilizar el posicionamiento dentro de una pantalla para la realización de pruebas, así como reconocimiento de texto por OCR.

SELENIUM IDE: La herramienta Selenium IDE, es un framework para pruebas de aplicaciones Web, descargable de forma gratuita desde su sitio web. Proporciona una herramienta de grabación y playback, que permite desarrollar pruebas sin necesidad de aprender un lenguaje de Scripting.

Incluye características como grabación, playback, selección de campos, autocompletar formularios, pruebas de recorrido (Walkthrough), debug, puntos de control, scripts Ruby y otros formatos.

MICROSOFT TEST MANAGER: Es la herramienta propiedad de Microsoft para la gestión y automatización de pruebas. Esta herramienta está incluida en Microsoft Visual Studio Ultimate 2010 o en Visual Studio Test Professional 2010.

El interfaz y el código generado en los scripts son bastante intuitivo, se debe de integrar con Team Foundation Server que almacena los casos de prueba y requerimientos entre otras cosas. El código generado se llama coded UI que graba operaciones de interfaz basado en Visual C#.NET.

Además, se pueden ejecutar las pruebas automáticas tanto en máquinas virtuales como físicas. Se instala en sistemas operativos Windows".

Dicho esto, y analizando las herramientas comparadas y dadas las características de los

aplicativos los cuales son: el aplicativo seleccionado es web, contiene diversas

funcionalidades y flujos repetitivos soportados en diferentes navegadores, por lo cual de acuerdo al cuadro comparativo nos ayuda para nuestro proceso de automatización de pruebas funcionales, la herramienta Selenium IDE, la cual trabaja con aplicaciones web y soporta diversos navegadores lo cual es lo principal conjuntamente se puede trabajar con diferentes plataformas, además tiene como funcionalidad principal grabar cada paso del funcionamiento y replicarlo con los scripts construidos, además de ser intuitivo para usar ya que nos ofrece realizar la automatización con grabaciones de pasos realizados y que tienen un tiempo de demora menor que las otras herramientas.

Selenium IDE se envía con una amplia estructura de flujo de control, con comandos disponibles como if, while y times, los cuales son fáciles de programar por el tipo de lenguaje que se está utilizando. Tiene una ventaja grande al ser una herramienta opensource que nos permite gozar de sus beneficios.

Comenzar con Selenium IDE no requiere ninguna configuración adicional, aparte de instalar la extensión en su navegador. Proporciona una herramienta fácil de usar que brinde una respuesta instantánea. Cuanto más fácil sea hacerlo, es más probable que las personas realicen pruebas, lo que a su vez da como resultado aplicaciones mejor probadas.

CARACTERISTICAS	UNIFIED FUNCTIONAL TESTING	SELENIUM IDE	MICROSOFT TEST MANAGER
Simulación equivalente a la acción del usuario final.	OK	OK	OK
Administración y almacenamiento de objetos.	OK	OK	OK
Soporte para distintos navegadores.	OK	OK	OK
Parámetros de reconocimiento de objetos.	OK	OK	OK
Integración con herramientas de gestión de pruebas	OK	OK	OK
Numerosas tecnologías soportadas.	OK	OK, sólo web.	OK
Soporte para Sistema operativo / plataformas	Windows	Windows, MacOS y Linux	Windows
Facilidad de creación de scripts.	OK	OK	OK
Lenguajes soportados para la creación de scripts.	Visual Basic	Numerosos lenguajes	Numerosos lenguajes

Tabla Comparativa de Herramientas para Automatización de Pruebas 1

CAPÍTULO V: APORTE PRÁCTICO

Se trabajará con el aplicativo desarrollado en sistema web llamado "Sistema Web de Cartas de Garantía" por la empresa consultora de software cuya principal funcionalidad es crear usuarios con diferentes roles y creación de cartas de garantía que son las atenciones médicas de los empleados de un seguro médico.

Implementaremos los pasos descritos en el proceso propuesto, para validar los puntos descritos en el proceso.

Creación de la Lista de Requerimientos: Nos apoyaremos con la herramienta Opensource llamada Testlink donde creamos juntamente con el cliente la Lista de Especificación de los Requisitos.

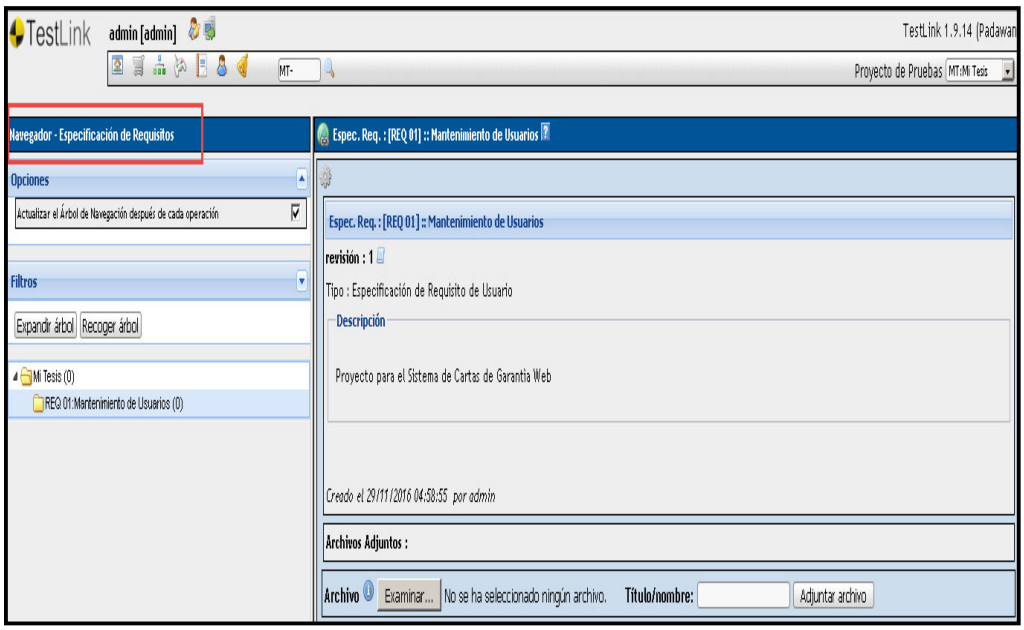


Figura 15. Especificación de Requisitos [Elaboración Propia].

Elaboración de los casos de Prueba:

Se especifica uno a uno los casos de pruebas a validar, para este desarrollo, así mismo se tendrá un histórico de los casos de pruebas que se están ejecutando. Se elaboraron 2 suites de pruebas para los dos módulos que tiene el sistema.

a. Suite 1: Mantenimiento de Usuarios: 6 casos definidos.

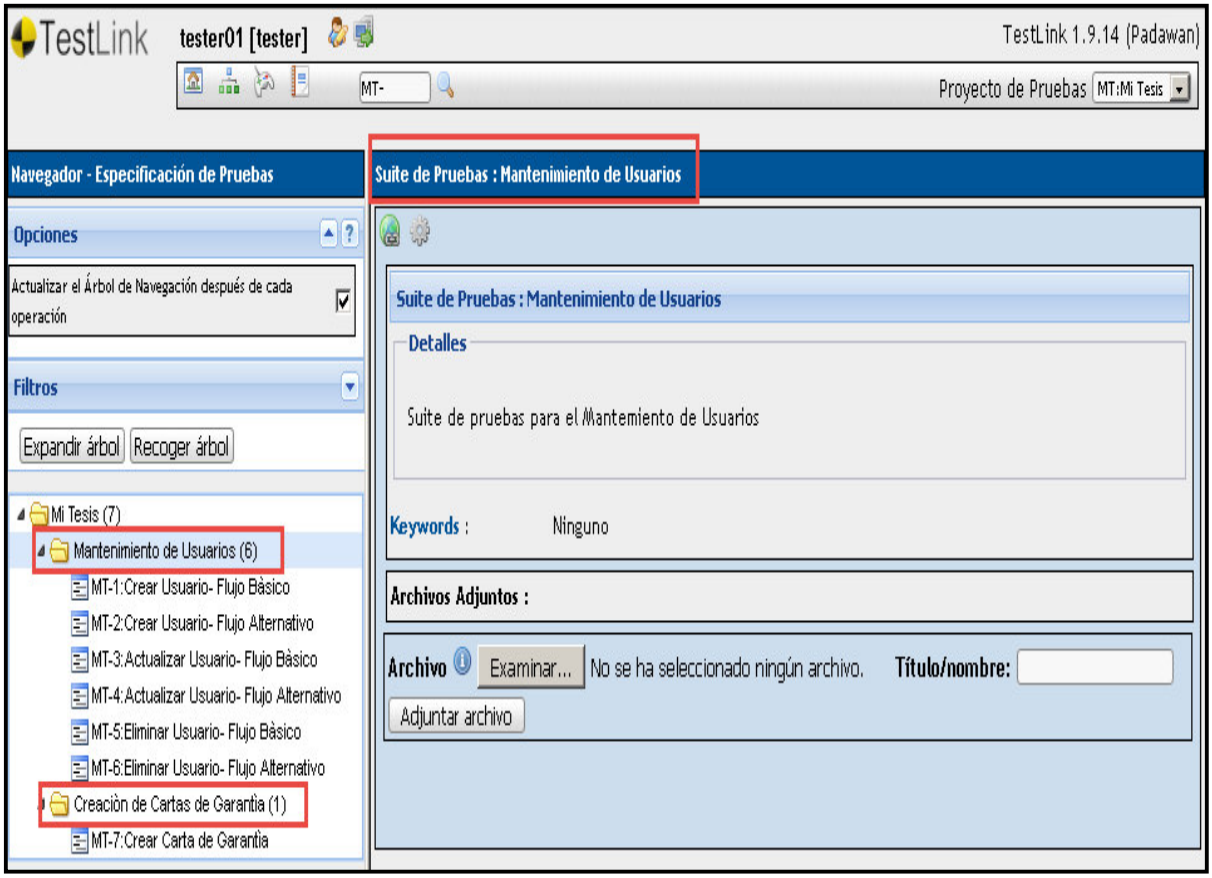


Figura 16. Suite de Pruebas: Mantenimiento de Usuarios [Elaboración Propia].

- Especificación de Caso de Prueba 1: Registrar Usuario- Flujo Básico.
  Se describe paso a paso el flujo de éxito al registrar un nuevo usuario.

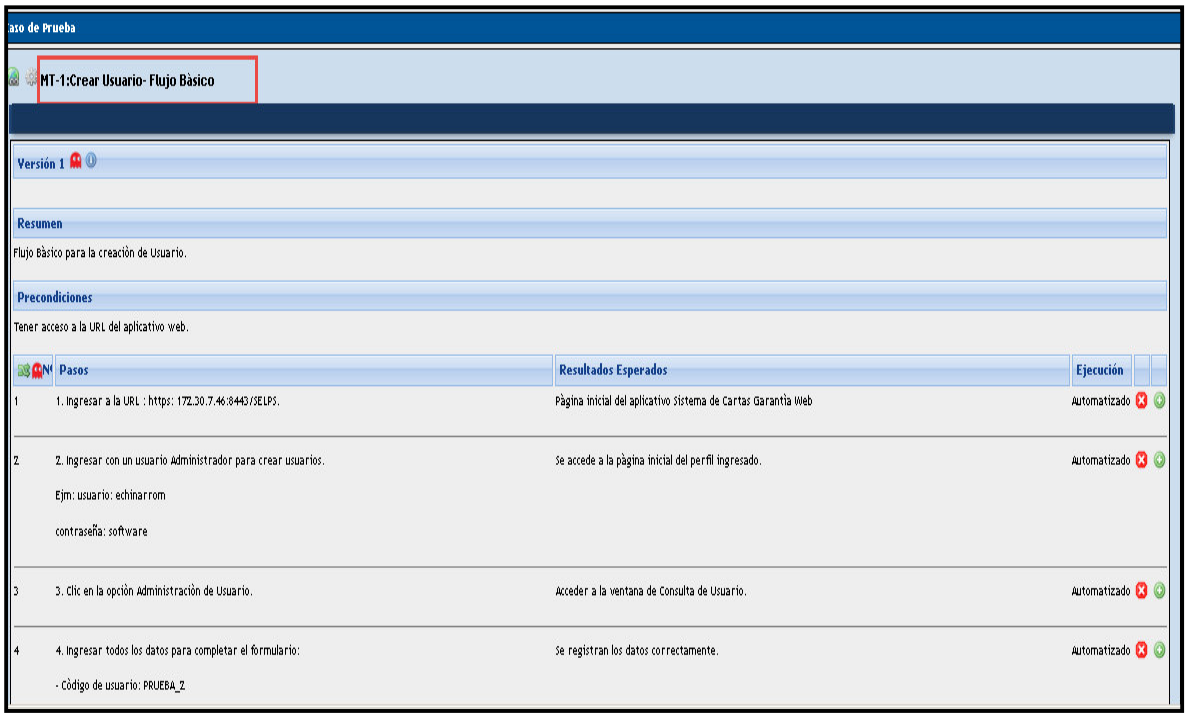


Figura 17. Especificación Caso de Prueba 1 [Elaboración Propia].

- Especificación de Caso de Prueba 2: Registrar Usuario- Flujo Alternativo.

  Definimos cada paso seguido al crear un nuevo usuario ya registrado anteriormente y que el sistema nos muestre un mensaje de error por querer agregar como nuevo usuario a uno ya registrado.

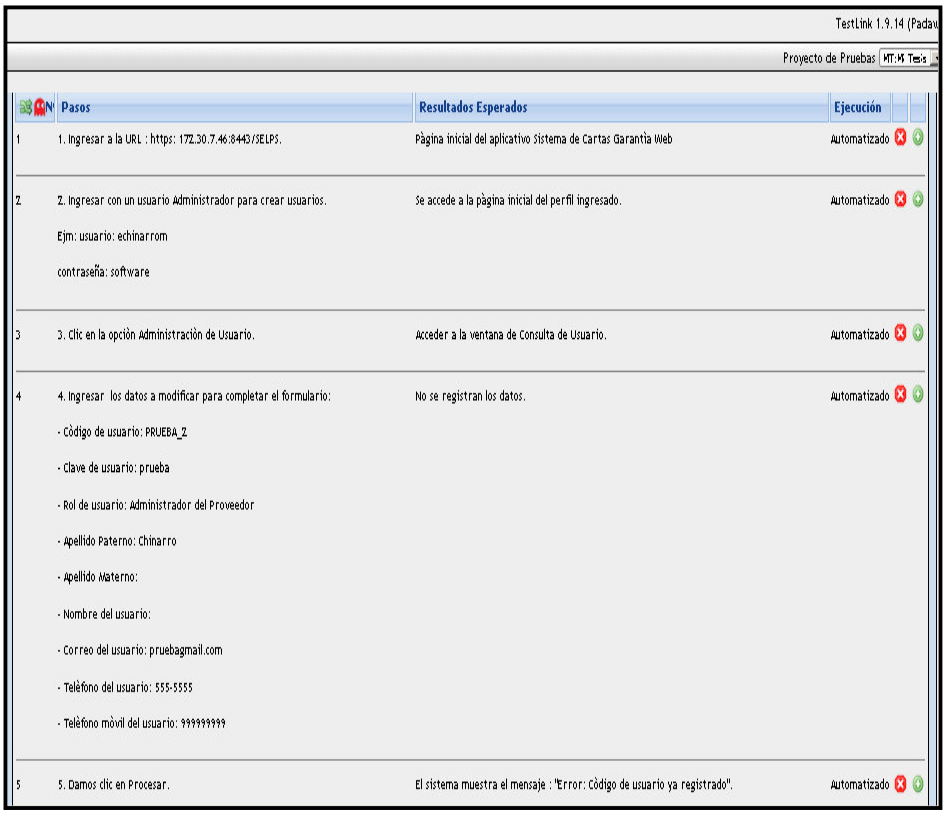
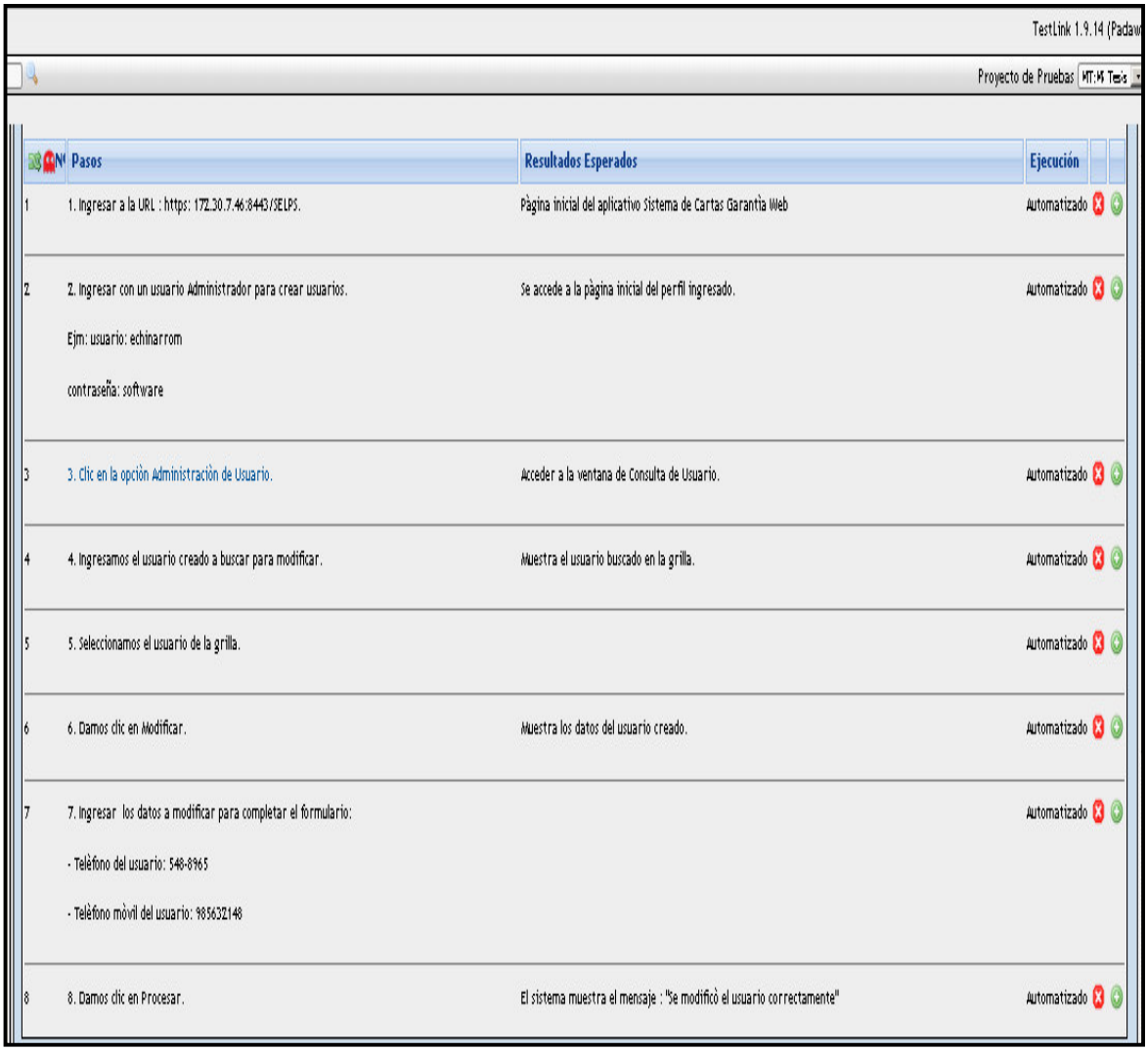


Figura 18. Especificación Caso de Prueba 2[Elaboración Propia].

- Especificación de Caso de Prueba 3: Actualizar Usuario- Flujo Básico

  Se describe el flujo de pasos seguidos para actualizar un usuario previamente registrado en el sistema.



TestLink 1.9.14 (Padaw

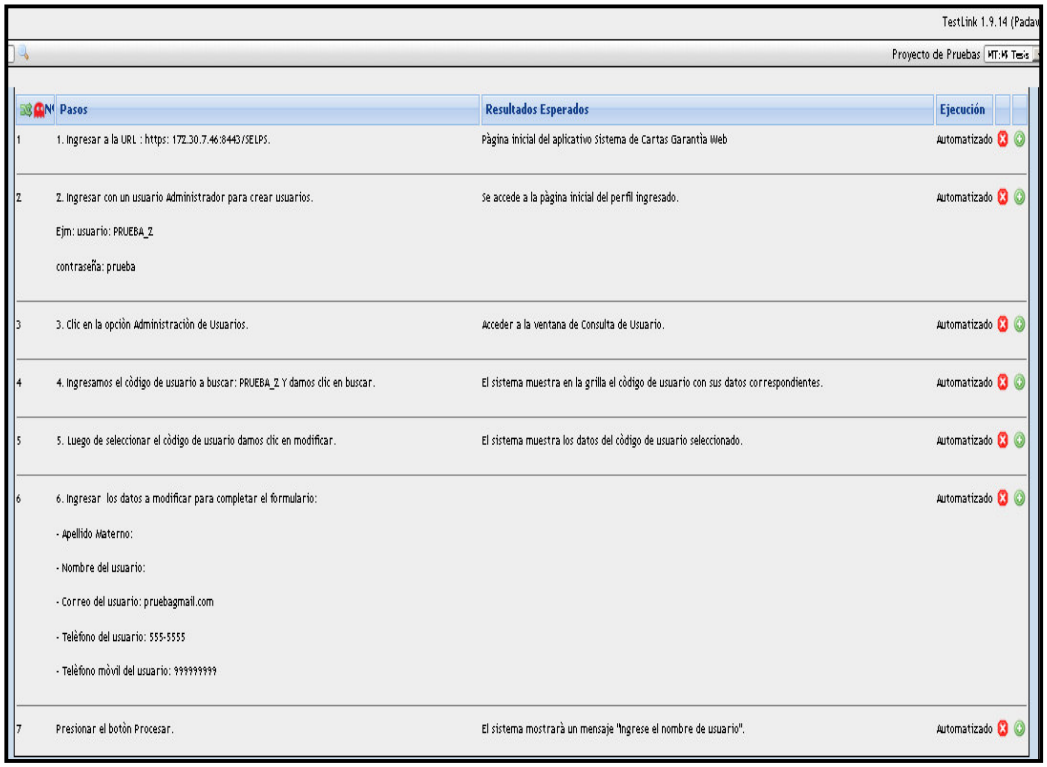
Proyecto de Pruebas MT:M Tests

	Pasos	Resultados Esperados	Ejecución
1	1. Ingresar a la URL : https: 172.30.7.46:8443/SELPS.	Pàgina inicial del aplicativo Sistema de Cartas Garantìa Web	Automatizado
2	2. Ingresar con un usuario Administrador para crear usuarios. Ejm: usuario: echinarrom contraseña: software	Se accede a la pàgina inicial del perfil ingresado.	Automatizado
3	3. Clic en la opciòn Administraciòn de Usuario.	Acceder a la ventana de Consulta de Usuario.	Automatizado
4	4. Ingresamos el usuario creado a buscar para modificar.	Muestra el usuario buscado en la grilla.	Automatizado
5	5. Seleccionamos el usuario de la grilla.		Automatizado
6	6. Damos clic en Modificar.	Muestra los datos del usuario creado.	Automatizado
7	7. Ingresar los datos a modificar para completar el formulario: - Telèfono del usuario: 548-8965 - Telèfono mòvil del usuario: 985632148		Automatizado
8	8. Damos clic en Procesar.	El sistema muestra el mensaje : "Se modificò el usuario correctamente"	Automatizado

Figura 19. Especificación Caso de Prueba 3[Elaboración Propia].

- Especificación de Caso de Prueba 4: Actualizar Usuario- Flujo Alternativo

  Se define paso a paso el flujo que se sigue al actualizar un usuario ya registrado en el sistema, pero dejando algún campo obligatorio vacío.



TestLink 1.9.14 (Pada

Proyecto de Pruebas: MT:Mi Tesis

	Pasos	Resultados Esperados	Ejecución
1	1. Ingresar a la URL : https: 172.30.7.46:8443/SELPS.	Pàgina inicial del aplicativo Sistema de Cartas Garantìa Web	Automatizado
2	2. Ingresar con un usuario Administrador para crear usuarios. Ejm: usuario: PRUEBA_2 contraseña: prueba	Se accede a la pàgina inicial del perfil ingresado.	Automatizado
3	3. Clic en la opciòn Administraciòn de Usuarios.	Acceder a la ventana de Consulta de Usuario.	Automatizado
4	4. Ingresamos el còdigo de usuario a buscar: PRUEBA_2 Y damos clic en buscar.	El sistema muestra en la grilla el còdigo de usuario con sus datos correspondientes.	Automatizado
5	5. Luego de seleccionar el còdigo de usuario damos clic en modificar.	El sistema muestra los datos del còdigo de usuario seleccionado.	Automatizado
6	6. Ingresar los datos a modificar para completar el formulario: - Apellido Materno: - Nombre del usuario: - Correo del usuario: pruebagmail.com - Teléfono del usuario: 555-5555 - Teléfono móvil del usuario: 999999999		Automatizado
7	Presionar el botòn Procesar.	El sistema mostrarà un mensaje "Ingrese el nombre de usuario".	Automatizado

Figura 20. Especificación Caso de Prueba 4 [Elaboración Propia].

- Especificación de Caso de Prueba 5: Eliminar Usuario- Flujo Básico

  Definimos los pasos seguidos al eliminar un usuario registrado previamente en el sistema.

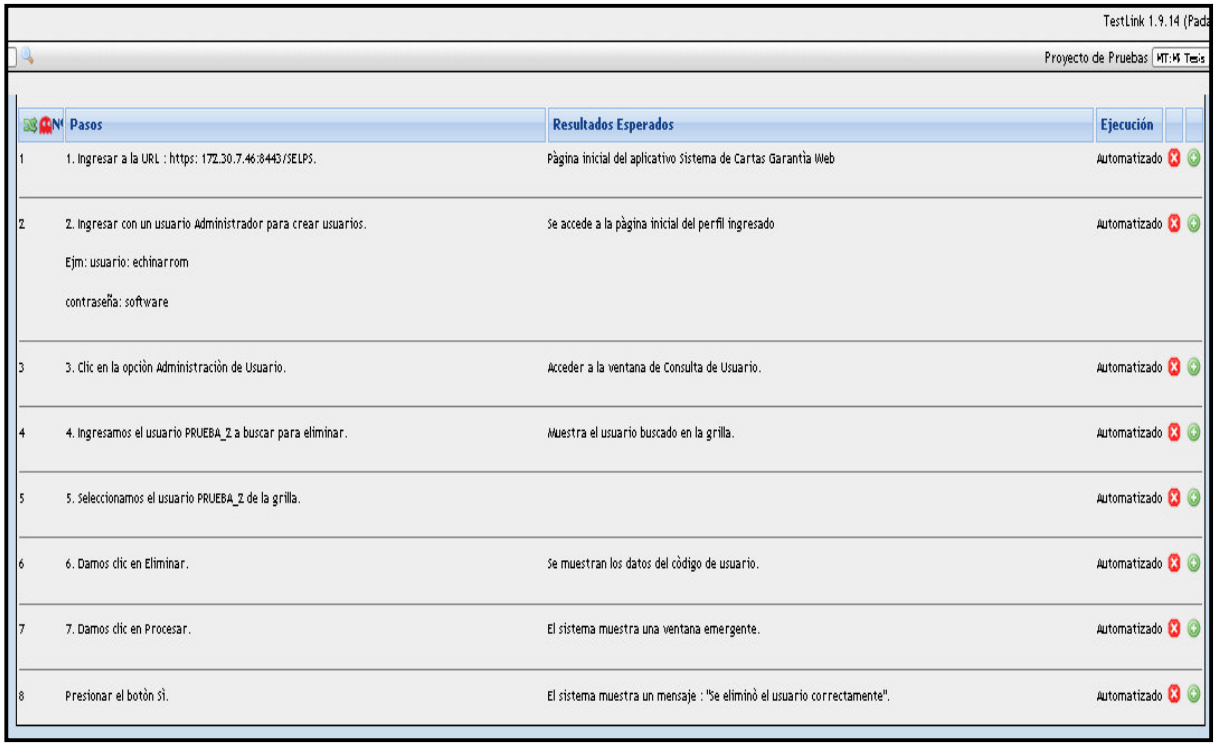


Figura 21. Especificación Caso de Prueba 5 [Elaboración Propia].

- Especificación de Caso de Prueba 6: Eliminar Usuario- Flujo Alternativo: Se sigue paso a paso el flujo al eliminar un usuario que no se encuentra en el sistema o ya ha sido eliminado anteriormente.

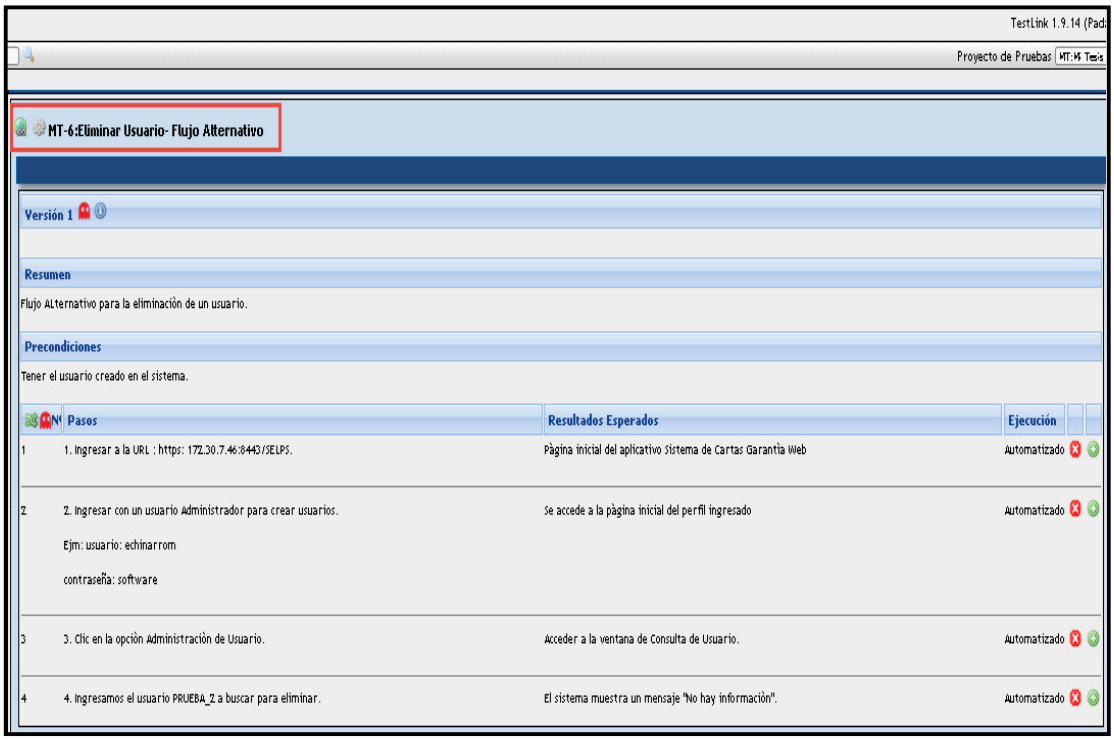


Figura 22. Especificación Caso de Prueba 6 [Elaboración Propia].

- Suite 2: Creación de Cartas de Garantía: 1 caso de prueba definido.

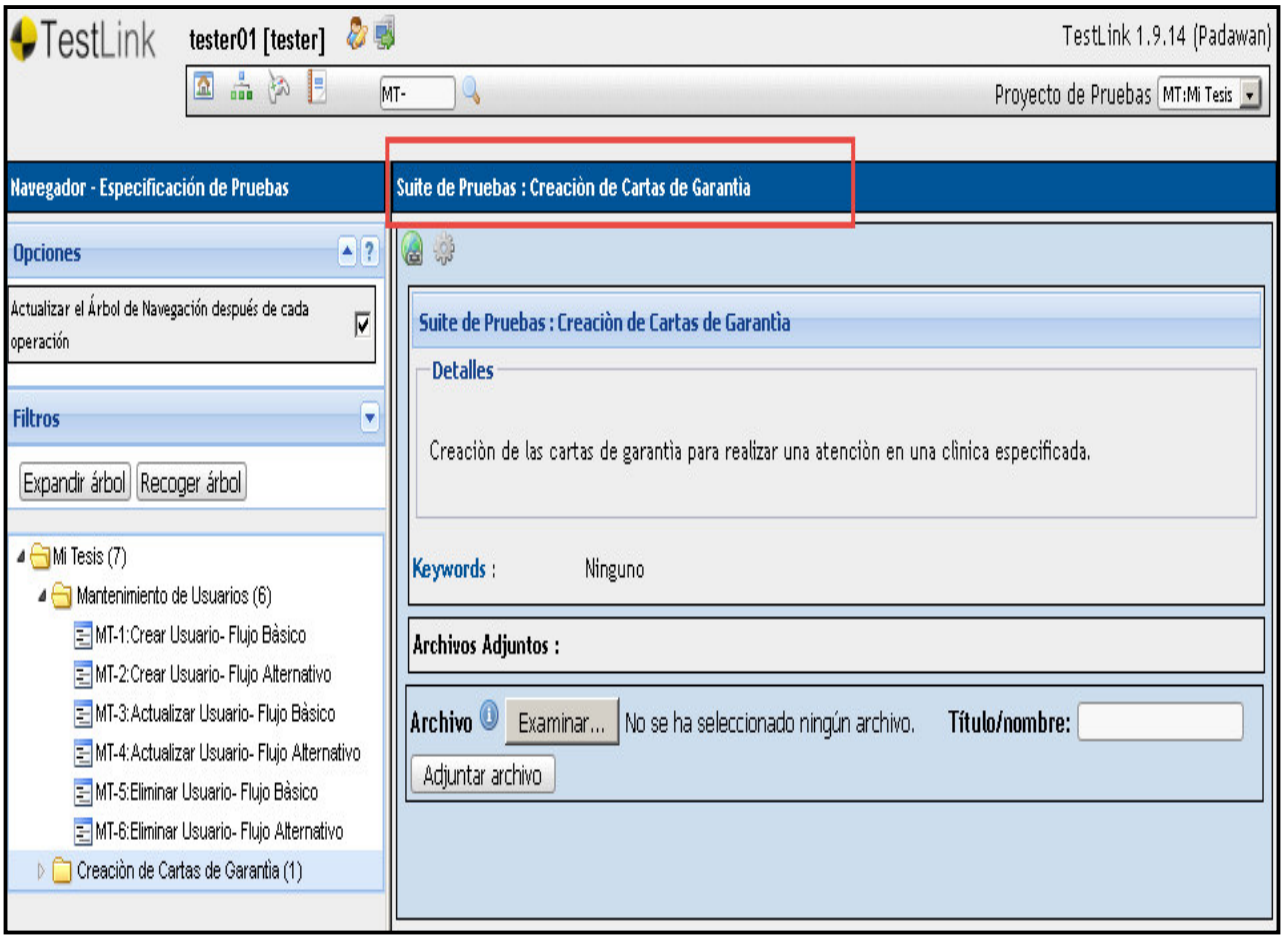
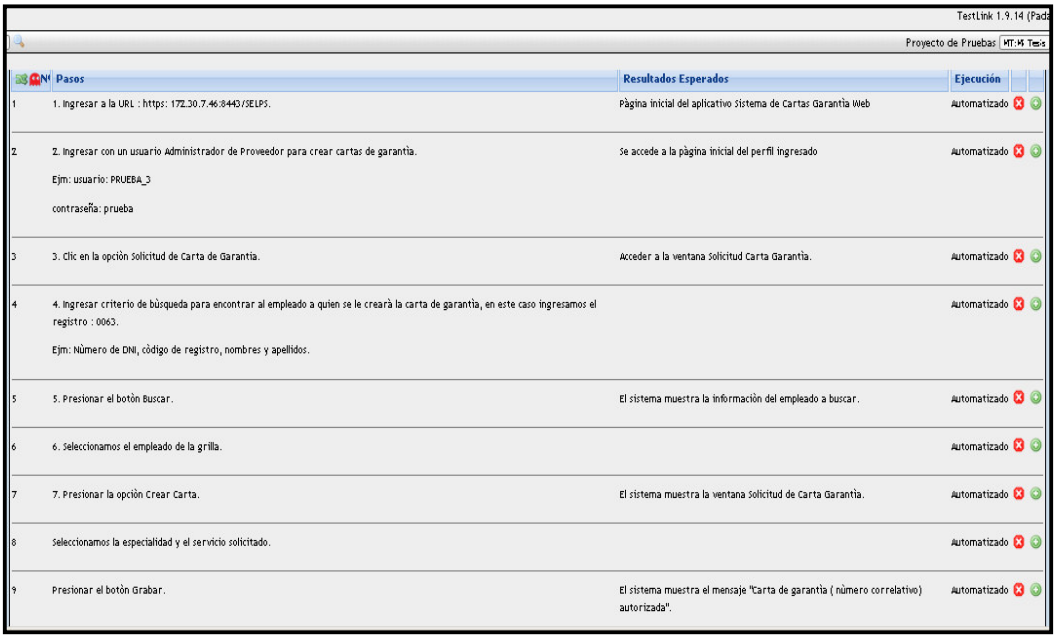


Figura 23. Suite de Prueba 2: Creación Carta de Garantía [Elaboración Propia].

- Especificación de Caso de Prueba 1: Crear Carta de Garantía



	Pasos	Resultados Esperados	Ejecución
1	1. Ingresar a la URL : https: 172.30.7.46:8443/SELPS.	Pàgina inicial del aplicativo Sistema de Cartas Garantìa Web	Automatizado
2	2. Ingresar con un usuario Administrador de Proveedor para crear cartas de garantìa. Ejm: usuario: PRUEBA_3 contraseña: prueba	Se accede a la pàgina inicial del perfil ingresado	Automatizado
3	3. Clic en la opciòn Solicitud de Carta de Garantìa.	Acceder a la ventana Solicitud Carta Garantìa.	Automatizado
4	4. Ingresar criterio de bùsqueda para encontrar al empleado a quien se le crearà la carta de garantìa, en este caso ingresamos el registro : 0063. Ejm: Nùmero de DNI, còdigo de registro, nombres y apellidos.		Automatizado
5	5. Presionar el botòn Buscar.	El sistema muestra la informaciòn del empleado a buscar.	Automatizado
6	6. Seleccionamos el empleado de la grilla.		Automatizado
7	7. Presionar la opciòn Crear Carta.	El sistema muestra la ventana Solicitud de Carta Garantìa.	Automatizado
8	Seleccionamos la especialidad y el servicio solicitado.		Automatizado
9	Presionar el botòn Grabar.	El sistema muestra el mensaje "Carta de garantìa (nùmero correlativo) autorizada".	Automatizado

Figura 24. Especificación Caso de Prueba 7[Elaboración Propia].

Creación del Plan de Pruebas: Se genera un Plan de Pruebas para definir los casos de pruebas a desarrollar y que dichos casos de prueba validen los requerimientos especificados por el cliente.

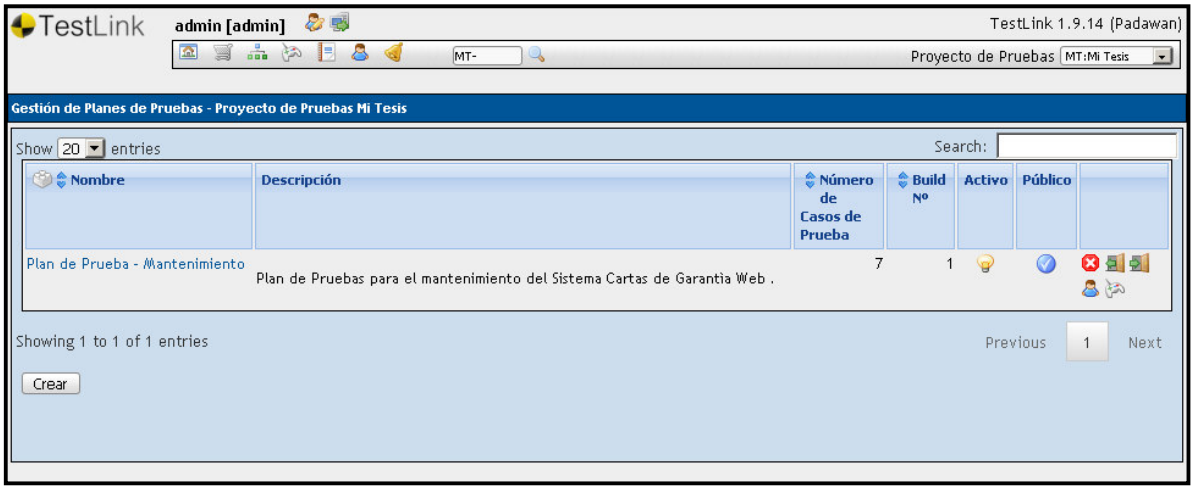


Figura 25. Creación Plan de Pruebas [Elaboración Propia].

Ejecución de los Casos de Prueba: Para este paso se usará el proceso de automatización de pruebas funcionales usando la herramienta Selenium:

- Ejecución caso de prueba 1: Registrar usuario – Flujo Básico.

  Se siguen uno a uno los pasos necesarios para registrar un nuevo usuario en el sistema, completando los todos los campos del formulario tal como se describe en la especificación del caso de prueba.

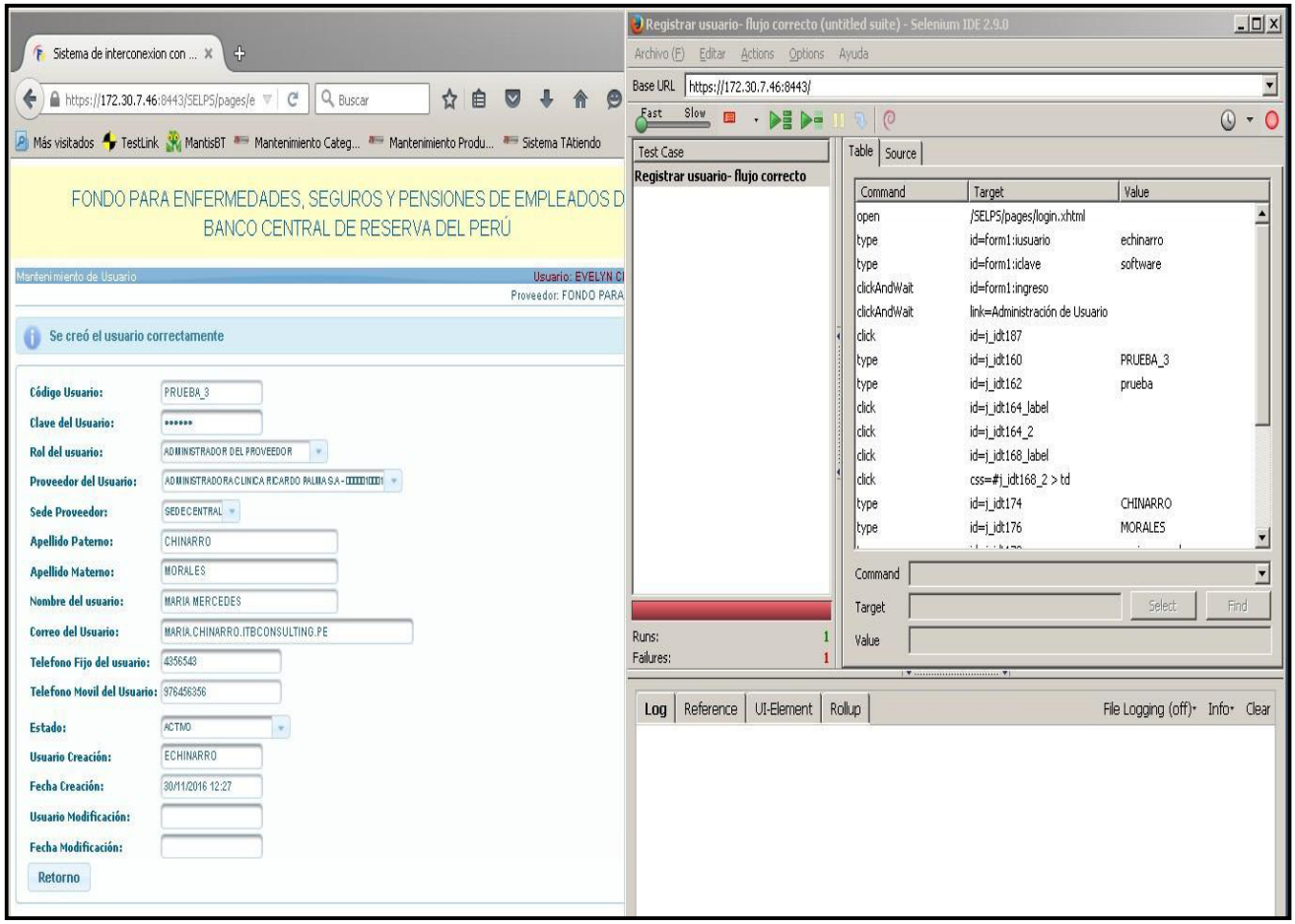


Figura 26. Ejecución caso de prueba 1 [Elaboración Propia].

- Ejecución caso de prueba 2: Registrar usuario – Flujo Alternativo.

  Para este caso de prueba tenemos que registrar los datos de un usuario ya creado previamente en el sistema para obtener un mensaje del sistema que nos mencione: “Error: Código de Usuario ya registrado”.

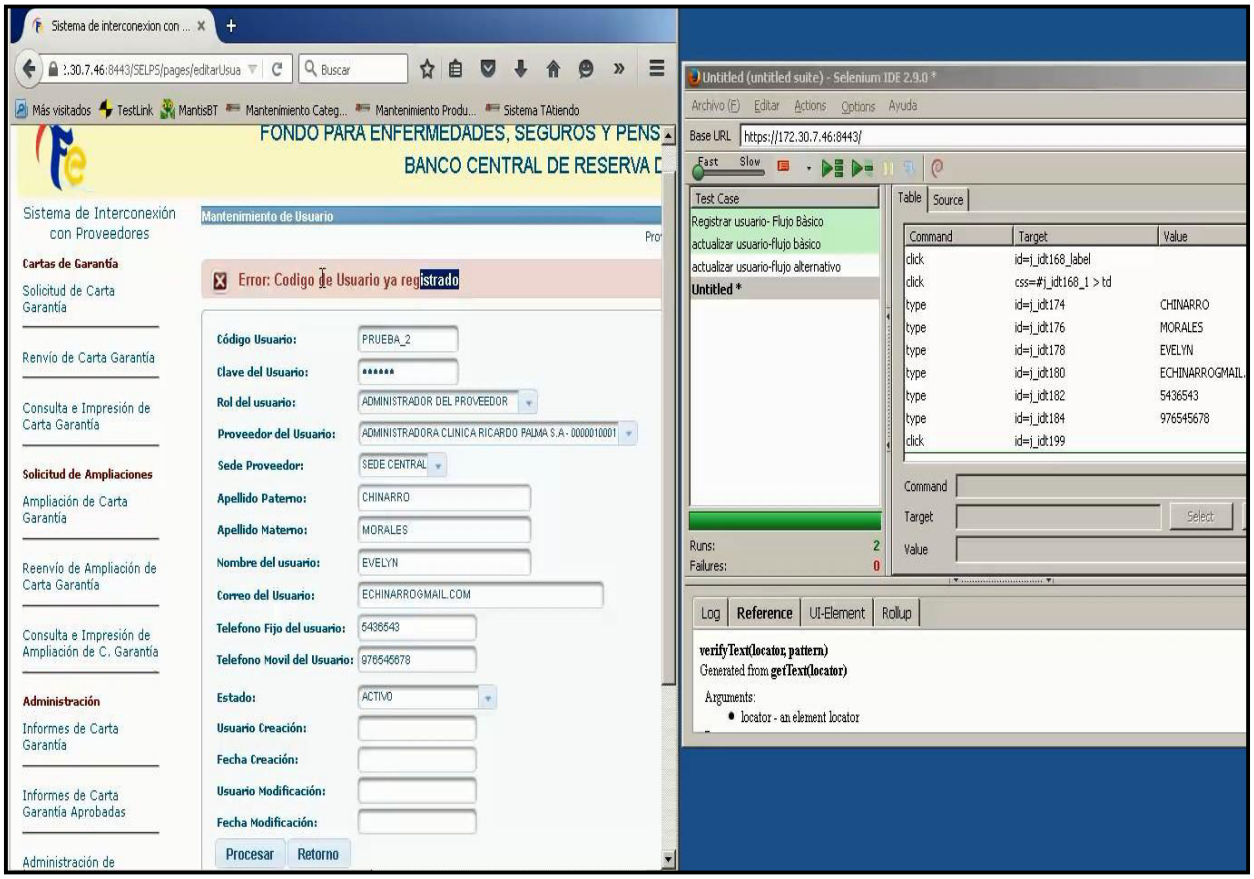


Figura 27. Ejecución caso de prueba 2 [Elaboración Propia].

- Ejecución caso de prueba 3: Actualizar usuario – Flujo Básico

  Previamente haber registrado a un usuario en el sistema, buscamos el código de usuario registrado y lo seleccionamos para modificar sus datos.

  Luego de modificar los datos del usuario registrado, el sistema te muestra un mensaje: “Se modificó el usuario correctamente”.

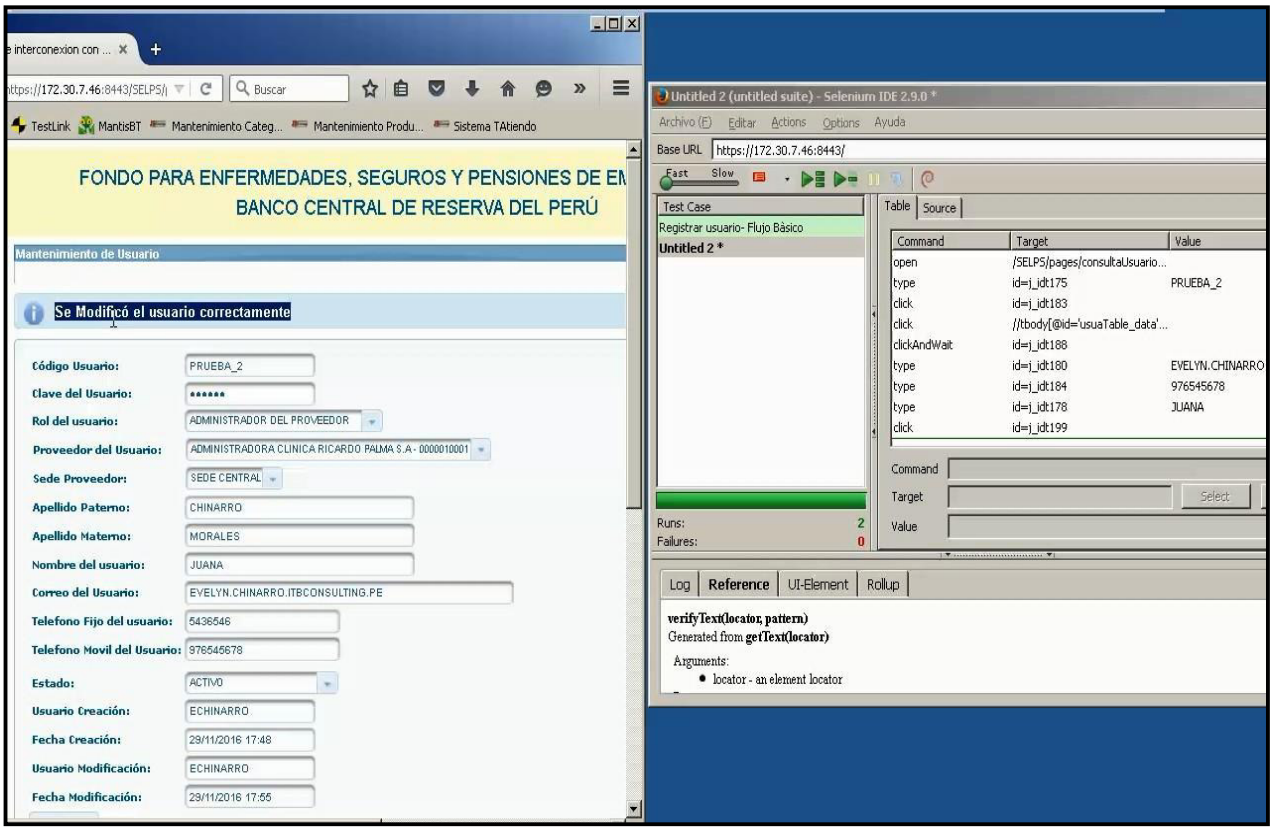


Figura 28. Ejecución caso de prueba 3 [Elaboración Propia].

- Ejecución caso de prueba 4: Actualizar usuario – Flujo Alternativo.

  Al querer modificar los datos de un usuario ya registrado, no colocamos todos los campos obligatorios y lo dejamos en blanco, el sistema me muestra un mensaje: "Ingrese el Nombre del usuario".

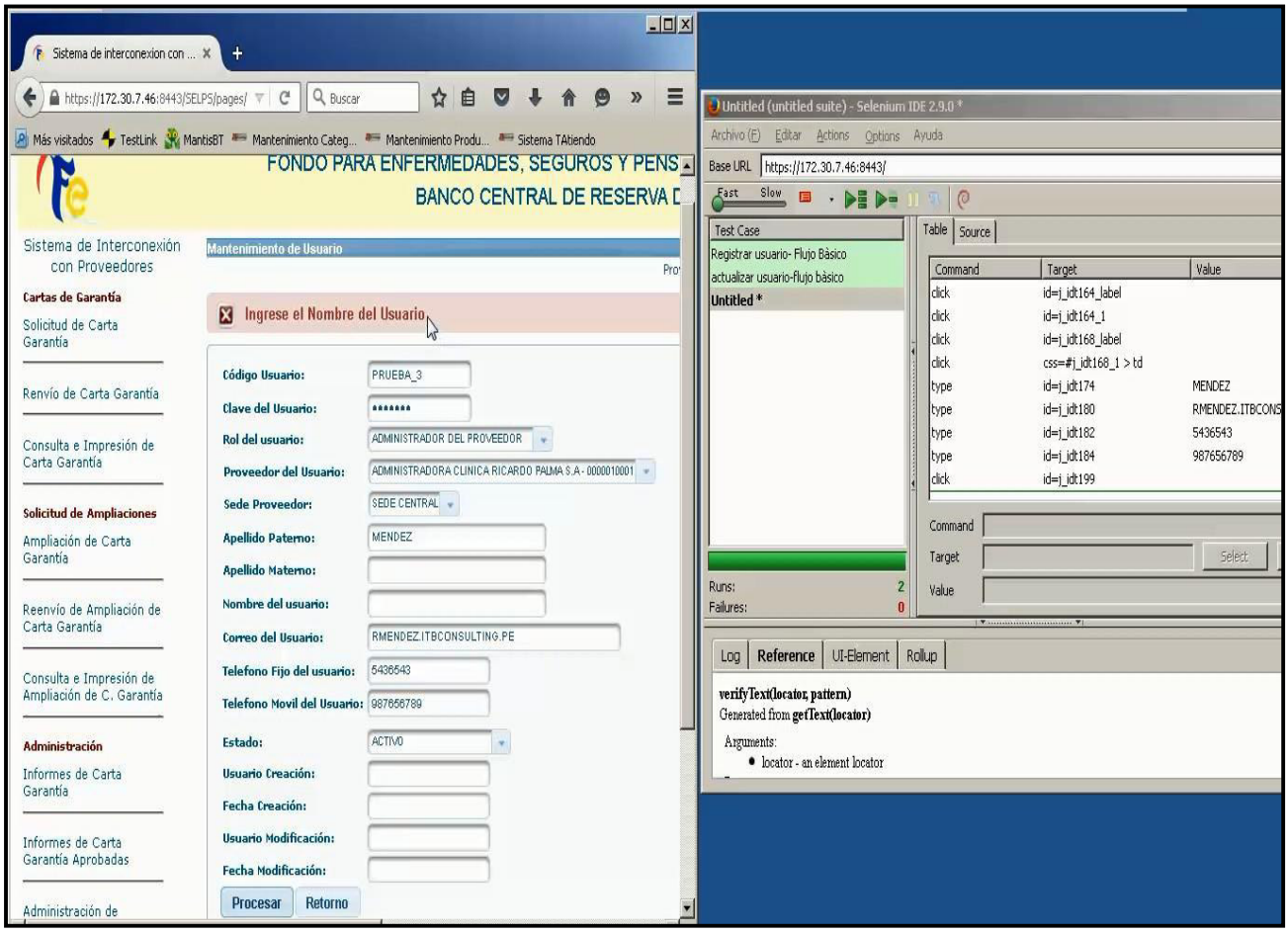


Figura 29. Ejecución caso de prueba 4 [Elaboración Propia].

- Ejecución caso de prueba 5: Eliminar usuario – Flujo Básico.

  Para eliminar un usuario ya registrado en el sistema, primero lo buscamos y damos clic en el botón eliminar, el sistema me muestra los datos del usuario.

  Procedemos a darle clic en el botón procesar, me muestra una ventana emergente donde me pide que confirme la eliminación del usuario.

  Finalmente me muestra un mensaje: “Se Eliminó el usuario correctamente”.

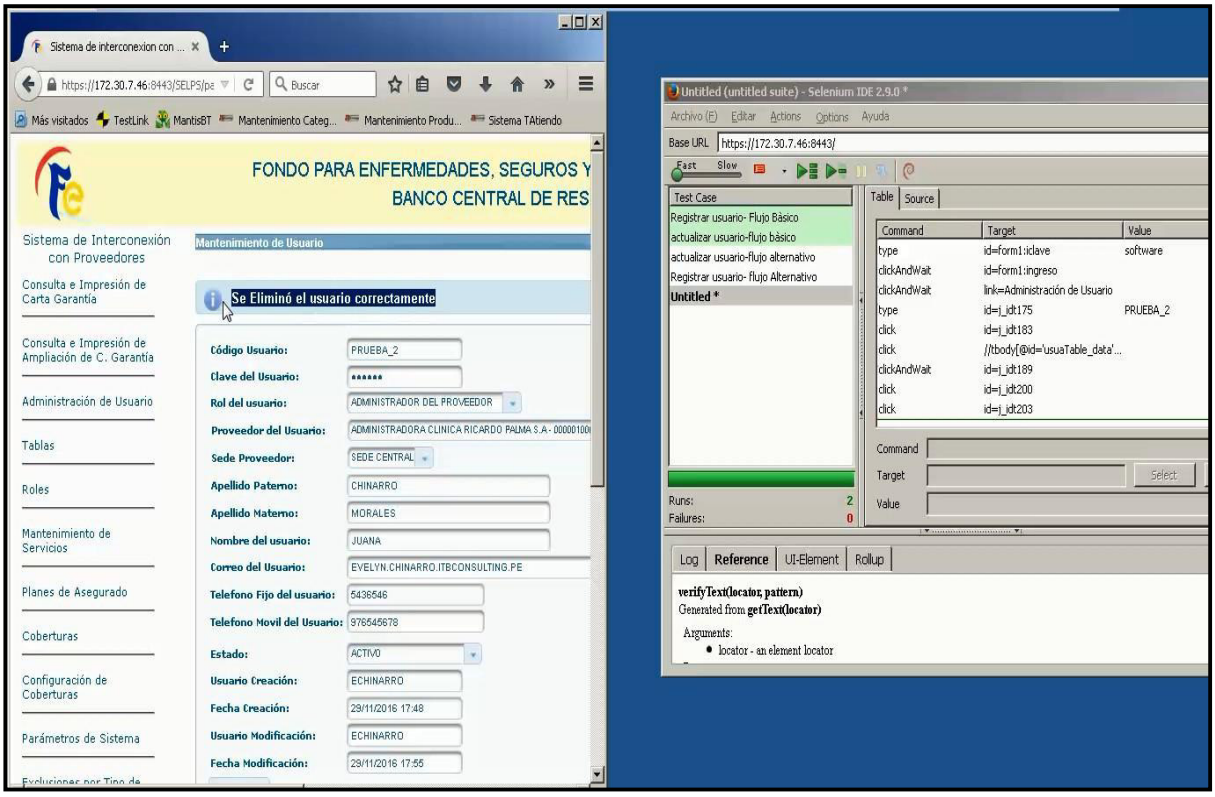


Figura 30. Ejecución caso de prueba 5 [Elaboración Propia].

- Ejecución caso de prueba 6: Eliminar usuario – Flujo Alternativo.

  Al buscar un usuario no registrado en el sistema o uno que ya haya sido eliminado, el sistema me mostrará un mensaje: “No hay coincidencias”.

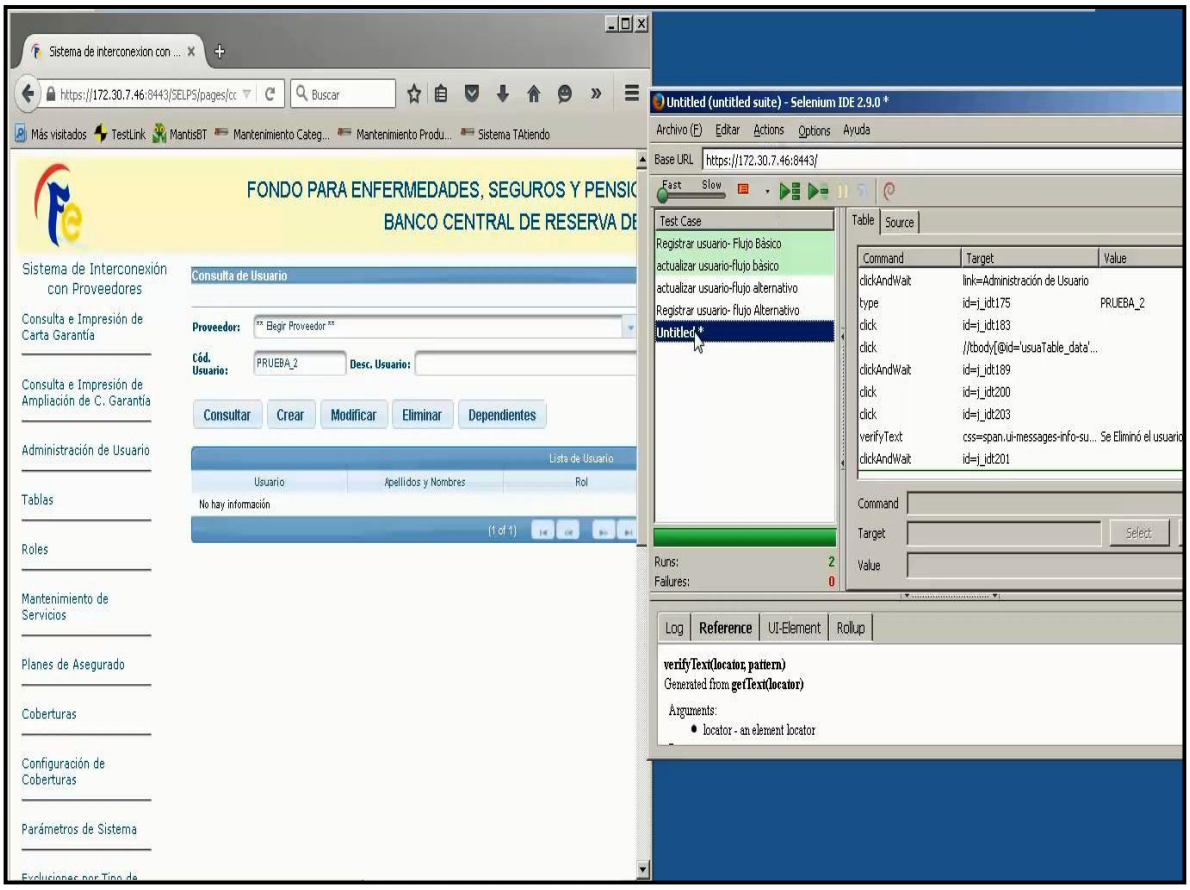


Figura 31. Ejecución caso de prueba 6 [Elaboración Propia].

- Ejecución caso de prueba 7: Crear carta de garantía.

  Se ejecuta el flujo descrito en la especificación del caso sobre la creación de la carta de garantía.

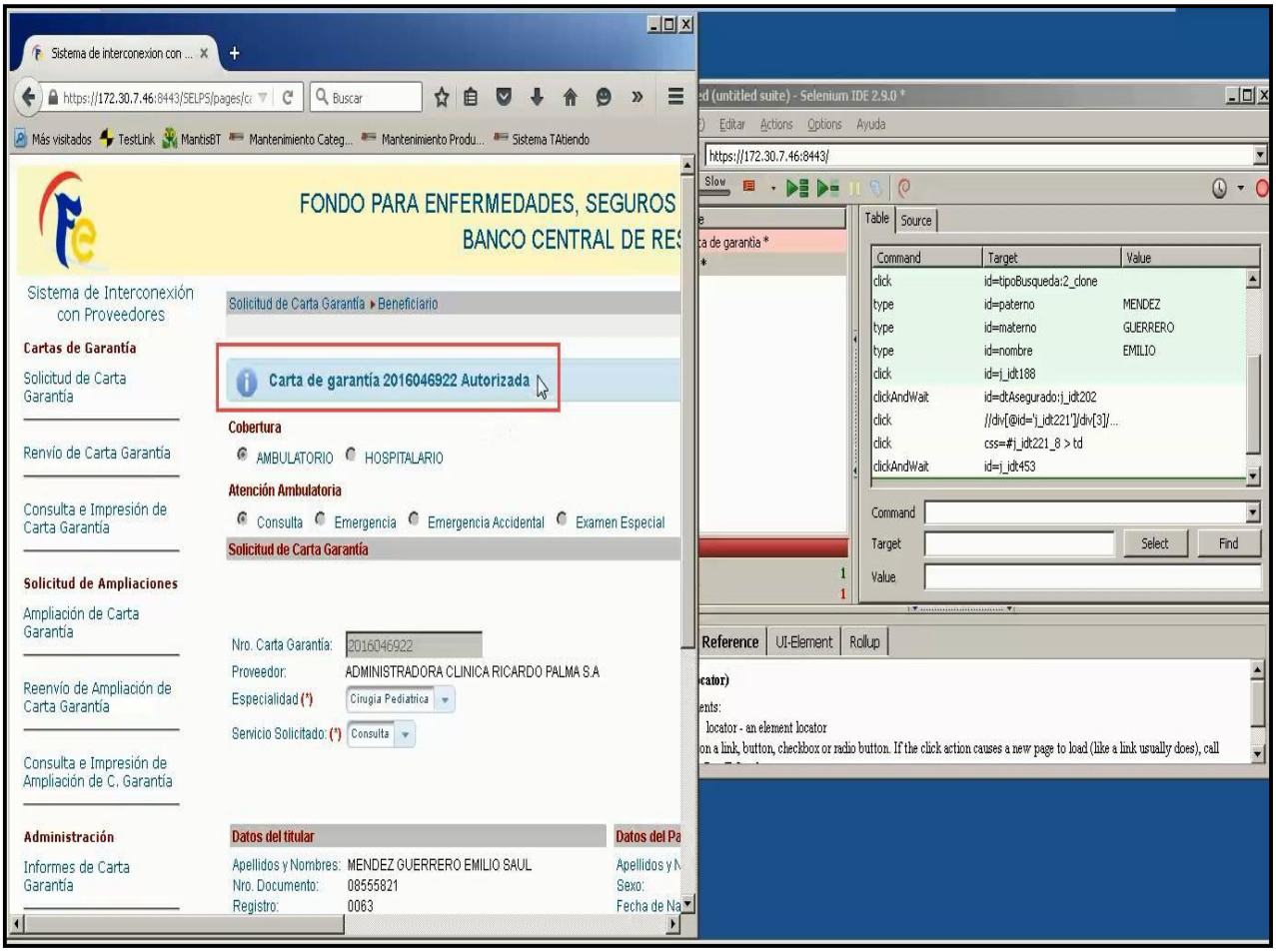
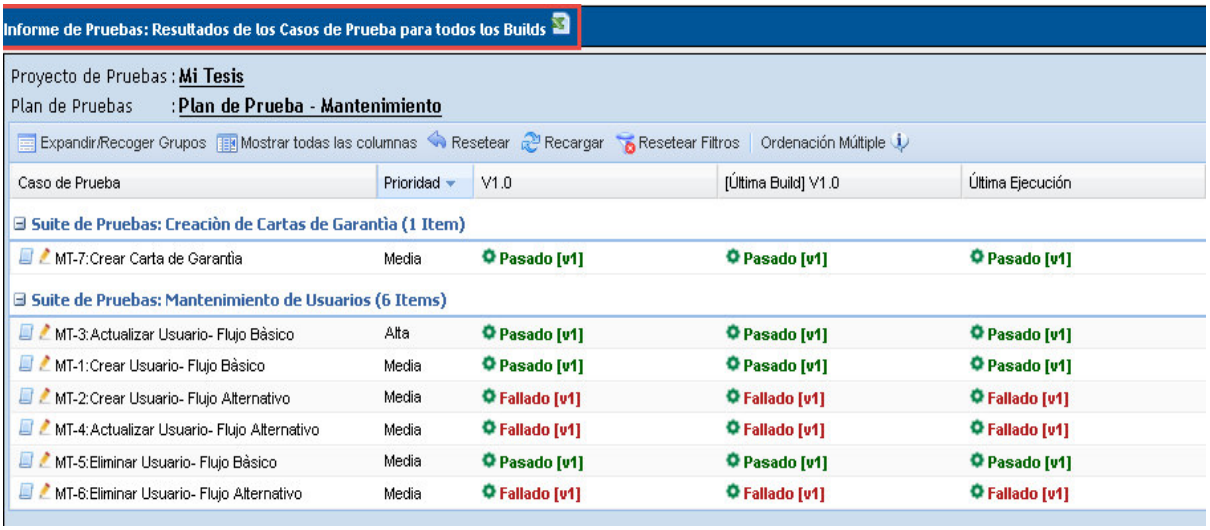


Figura 32. Ejecución caso de prueba 7 [Elaboración Propia].

VALIDACIÓN DEL PROCESO IMPLEMENTADO

Luego de la ejecución de los casos de pruebas como parte del proceso definido, aquí se muestran los resultados obtenidos en un reporte brindado por la aplicación utilizada para la documentación de los casos de pruebas.

En el reporte se muestra el estado de cada caso de prueba y el tiempo de ejecución, el cual comparado con el tiempo de ejecución para las pruebas que se realizaban antes de la implementación del proceso de pruebas elaborado, es mucho menor lo cual denota el aporte significativo que ha tenido el proceso elaborado en la organización.



Informe de Pruebas: Resultados de los Casos de Prueba para todos los Builds

Proyecto de Pruebas : **Mi Tesis**
Plan de Pruebas : **Plan de Prueba - Mantenimiento**

Expandir/Recoger Grupos | Mostrar todas las columnas | Resetear | Recargar | Resetear Filtros | Ordenación Múltiple

Caso de Prueba	Prioridad	V1.0	[Última Build] V1.0	Última Ejecución
Suite de Pruebas: Creaciòn de Cartas de Garantìa (1 Item)				
MT-7:Crear Carta de Garantìa	Media	Pasado [v1]	Pasado [v1]	Pasado [v1]
Suite de Pruebas: Mantenimiento de Usuarios (6 Items)				
MT-3:Actualizar Usuario- Flujo Bàsico	Alta	Pasado [v1]	Pasado [v1]	Pasado [v1]
MT-1:Crear Usuario- Flujo Bàsico	Media	Pasado [v1]	Pasado [v1]	Pasado [v1]
MT-2:Crear Usuario- Flujo Alternativo	Media	Fallado [v1]	Fallado [v1]	Fallado [v1]
MT-4:Actualizar Usuario- Flujo Alternativo	Media	Fallado [v1]	Fallado [v1]	Fallado [v1]
MT-5:Eliminar Usuario- Flujo Bàsico	Media	Pasado [v1]	Pasado [v1]	Pasado [v1]
MT-6:Eliminar Usuario- Flujo Alternativo	Media	Fallado [v1]	Fallado [v1]	Fallado [v1]

Figura 33. Resultados de la Ejecución [Elaboración Propia].

El proceso manual de pruebas de software utilizado por la organización al aplicativo según cronograma demoraba aproximadamente 3 semanas de ejecución de todos los casos de prueba descritos en el plan de pruebas. Al aplicar el nuevo proceso elaborado y definido se validaron los tiempos de ejecución con la herramienta de apoyo para la automatización de las pruebas, el cual disminuyó considerablemente ya que se ejecutó en exactamente 1 semana, dando tiempo para la realización de otras pruebas más minuciosas y así ofrecerle al usuario un producto de calidad.

Como se ve en la imagen, se obtienen gráficos que visualizan las métricas obtenidas luego de la ejecución del proceso elaborado. Estos reportes apoyan en la presentación de los informes que se presentan al usuario final, donde se mapean los casos fallados y pasados luego del final de la ejecución del proceso.

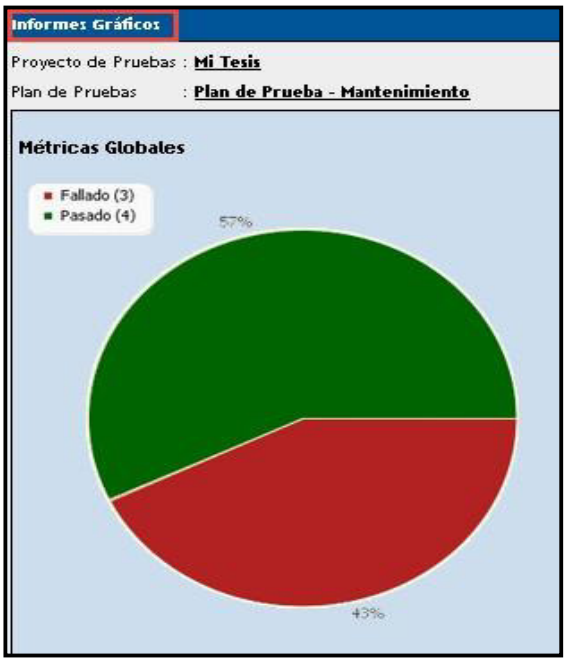


Figura 34. Gráfica de Ejecución [Elaboración Propia].

CAPÍTULO VI: CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES:

- Se definió e implementó un proceso de pruebas funcionales, el cual se le proporcionó a la empresa consultora para así garantizar un mejor desempeño durante el proceso de las pruebas funcionales, entregando un producto con mejor calidad.
- Se documentaron los casos ejecutados, los cuales ayudaron a contemplar más escenarios de pruebas de acuerdo con lo especificado por el cliente en sus requerimientos iniciales.
- Se evaluó y seleccionó a la herramienta que aportaba mejor con sus características al proceso de automatización.
- El proceso incidió positivamente en la disminución del tiempo requerido por las pruebas funcionales, además se utilizó una herramienta de automatización para su apoyo en el proceso definido.

RECOMENDACIONES:

- Tener en cuenta que el proyecto presentado está orientado a las pequeñas empresas que están surgiendo en el mundo del TI.

- La presente tesis está enfocada en el desarrollo de proyectos nuevos, pero también se puede orientar y mejorar para los futuros mantenimientos de este.

- Al comenzar a desarrollar el aplicativo se deben considerar las actividades necesarias para la fase de las pruebas y así brindar un aplicativo de calidad.

ANEXO 01

Documento: Lista de Requerimientos

GENERALIDADES

El objetivo de este documento es identificar los requerimientos del usuario y tener un detalle de estos para asegurar que todos los involucrados en el proyecto cuentan con un entendimiento de sus necesidades.

DESCRIPCIÓN DEL PROYECTO

DESCRIPCIÓN DE LA SITUACIÓN ACTUAL DEL SISTEMA

>> Llenar sólo en caso de que el presente proyecto de Mantenimiento, de lo contrario escribir "Proyecto de Desarrollo" o" Proyecto de Adquisición", según sea el caso. Incluir diagrama de contexto.

RELACIÓN CON OTROS SISTEMAS

>> Nombrar los sistemas y productos (existentes o futuros) que tendrán relación con el sistema a desarrollar. Se indicará si actúa como soporte de éste, si formara parte del mismo o serán sistemas externos.. De igual manera se especificarán las interfaces, si existen. En caso de ser un sistema existente, la relación con otros sistemas a definir tomará en cuenta los siguientes grupos definidos: Relaciones Actuales, Relaciones a implementar, y Relaciones por Eliminar.

CARACTERÍSTICAS – MODELO DEL PROCESO DEL NEGOCIO

>> Debe detallarse en forma descriptiva o gráfica el flujo del proceso en circuito completo, además de las relaciones con otros negocios. Explicar y resaltar el (los) proceso(s) en el cual(es) serán impactados por los cambios solicitados en la presente lista de requerimientos.

Ejemplo de Flujo del Proceso

DEFINICIÓN Y ELABORACIÓN DE REQUISITOS

Requisitos Funcionales (RF)

>> Indicar en la siguiente tabla los requisitos que se obtienen de la captación de la información obtenida del usuario y su entorno técnico; especificar lo que el sistema tiene que desarrollar, agrupando por funcionalidad, definiendo así el propósito y funcionalidad que el usuario requiere. Se debe identificar el impacto que tiene cada requerimiento considerando la criticidad de las necesidades a cubrir del usuario. A: Alto, M: Medio, B: Bajo.

Código Requisito	Impacto (A/M/B)	Descripción del Requisito
RF-01	**A**	**Título del RF**
Descripción: ***Detallar Requerimiento Funcional y las condiciones del producto a desarrollar.*** ***En el caso de ser una ventana se deberá especificar entradas y salidas de la ventana o dar el modelo de esta.***		
Excepciones: ***Indicar las condiciones que pueden generar variantes en la realización del requerimiento funcional o hacer referencia al RF que las contiene.***		
Entregables: ***Indicar salidas, campos o salidas intermedias del requerimiento funcional. De ser el caso, indicar anexo en el cual figure formatos de salida.*** *Anexo 1 - RF1.*		
Casuística de Validación: ***Especificar los condicionantes para verificar y validar el requisito, así como las restricciones sobre el modo en que éste será sometido a pruebas (funcionalidad, volumen, tiempo de respuesta, entre otros)***		

Requisitos de Interfaz (RI)

>> "En esta categoría se incluirán los requisitos relacionados con todos aquellos elementos externos con los cuales el producto o sistema interactuará. Estos elementos externos pueden ser sistemas operativos, paquetes estándar, software de otros sistemas, base de datos, entre otros. Se debe identificar el impacto que tiene cada requerimiento considerando la criticidad de las necesidades a cubrir del usuario. A: Alto, M: Medio, B: Bajo".

Código Requisito	Impacto (A/M/B)	RF(s) Asociado	Descripción del Requisito
RI-01	**A**	**RF-01**	**Título del RI**
*Descripción**: Detallar Requerimiento de Interfaz y las condiciones del producto a desarrollar.*** En el caso de ser una ventana se deberá especificar entradas y salidas de la ventana o dar el modelo de esta.			
RI-02	**M**	**RF-02**	**Título del RI**
*Descripción**: Detallar Requerimiento de Interfaz y las condiciones del producto a desarrollar.*** En el caso de ser una ventana se deberá especificar entradas y salidas de la ventana o dar el modelo de esta.			

Requisitos No Funcionales (RN)

>> Se refiere a las consideraciones y configuraciones mínimas que el sistema deberá soportar cuando se encuentre en funcionamiento, en cuanto a la definición de la arquitectura en la que se solicita el desarrollo, compatibilidad requerida, arquitectura de desarrollo deseada, plataforma, tiempos de respuesta requeridos, volumen de información a soportar, niveles de crecimiento, entre otros.

Código Requisito	Impacto (A/M/B)	Descripción del Requisito
RN-01	**A**	**Título del RN**
*Descripción**: Detallar Requerimiento No Funcional y las condiciones del producto a desarrollar.*** En el caso de ser una ventana se deberá especificar entradas y salidas de la ventana o dar el modelo de esta.		
RN-02		
*Descripción**: Detallar Requerimiento No Funcional y las condiciones del producto a desarrollar.*** En el caso de ser una ventana se deberá especificar entradas y salidas de la ventana o dar el modelo de esta.		

Requisitos de Seguridad (RS)

>> "Aplica para nuevos desarrollos o Mantenimiento Evolutivo / Adaptativo que requieran cambios en el módulo de seguridad. En esta categoría se identifican los requisitos necesarios para mantener la confidencialidad e integridad del sistema, así como la protección contra intrusos, alteración de datos e instalaciones. Adjunta ficha de requerimientos a cumplir".

Requisitos de Capacitación (RC)

>> En esta categoría se identifican los requisitos necesarios para que el usuario o usuarios finales estén en la capacidad de recibir el producto y empezar a utilizar las nuevas funcionalidades que éste le brinda. Se debe identificar el impacto que tiene cada requerimiento considerando la criticidad de las necesidades a cubrir del usuario. A: Alto, M: Medio, B: Bajo.

Código Requisito	Impacto (A/M/B)	Descripción del Requisito
RC-01	**A**	**Título del RC**
*Descripción**: Detallar Requerimiento de Capacitación y las condiciones del producto a desarrollar.***		
RC-02	**M**	**Título del RC**
*Descripción**: Detallar Requerimiento de Capacitación y las condiciones del producto a desarrollar.***		

Escenarios Operacionales

>> En esta categoría se indica una secuencia real de eventos o pasos de cómo el sistema responde a una solicitud, ya sea del propio sistema, de otro sistema o del usuario. Adicionalmente se debe incluir un Diagrama Escenarios en el que se detallan los actores y los escenarios operacionales relacionados.

Entradas: Se deben incluir los datos de entrada del escenario ya sea que correspondan a archivos o ventanas para ingreso de datos, así como el prototipo de ventanas de ser el caso. Salidas: Se deben incluir los datos de salida o resultados esperados del escenario ya sea que correspondan a archivos o ventanas, es decir, la información que el Sistema debe entregar.

Ej.: el requisito funcional es hacer un ingreso en línea de datos del cliente, como entrada tendríamos los datos ingresados por el cliente y como salida tendríamos Ventana de ingreso de datos, etc.

Estos requisitos corresponden al detalle de los requerimientos funcionales, de interfaces y demás que se identificaron en los puntos anteriores. Por tanto, no son trazables. >>

Código Escenario	Descripción del Escenario	Requisito asociado	Entradas	*Salidas*
EO-01	**Actores:** Persona o personas que interactúan en el proceso y que ejecutan o realizan la actividad. **Flujo Básico de Eventos:** Se describe la secuencia de actividades o pasos básicos, normales e invariables que se realizan en el escenario. **Flujo Alternativo de Eventos:** Describe la secuencia cuando el flujo básico de eventos se cumple	RF asociado al escenario operacional.	Una entrada puede tener más de una salida o no tener ninguna. En ese caso indicar 'No Aplica'	Una salida puede tener más de una entrada o no tener ninguna. En ese caso indicar 'No Aplica'

ANEXO 02

Documento: Matriz de Casos de Pruebas

Origen Front	Servicio o	Funcionalidad ad	Actor	Opción de Menú	Tipo Escenario	Código Caso de Prueba	Descripción Caso de Prueba	Resultado Esperado	Tablas Relacionadas	Fecha Planificada Testing	Fecha Real Testing	Estado (OK / NO OK / ANULADO)	Responsable Testing	Responsable Usuario	Responsable Firma	Comentario	Fecha Usuario	Firma

ANEXO 03

Documento: Matriz de Observaciones

NOMBRE PROYECTO

TAMAÑO (Pequeño/Media/Grande) :

COMPLEJIDAD (Baja/Media/Alta) :

TESTER RESPONSABLE :

Nro.	Fecha Real Testing	Responsable Desarrollo	Responsable Testing	Proyecto	Código Caso de Prueba	Descripción	Severidad	Tipificación del Error	Motivo de Devolución	Tipo de Prueba	Estado	Comentario	Fecha Reporte de Observación	Fecha de Solución	Total Rechazos	Respuesta Jefe de Proyecto
1																
2																

ANEXO 04

Documento: ACTA DE CAPACITACIÓN

< GTI-AAAA-NNM-SIGLA-Acta-Capacitación >[1]

<Este documento tiene el objetivo principal de formalizar la capacitación en el uso del sistema o aplicativos desarrollados en el proyecto. Firman el documento el Líder Usuario, Jefe de Proyecto, Jefe del área usuaria y Jefe del Dpto. de Gestión y Desarrollo de Soluciones de TI >.

Los firmantes dejan constancia de la capacitación en el uso de sistemas o aplicativos desarrollados en el proyecto "GTI-AAAA-NNM-SIGLA" realizada el DD de MMM de AAAA.

Líder Usuario Jefe de Proyecto

Jefe Área Usuaria Jefe DGDSTI

[1] En la nomenclatura del Acta de Capacitación del Proyecto AAAA = año de inicio del proyecto y NNN = número asignado al proyecto en el POI y SIGLA = descripción muy corta del nombre del proyecto asignada en el POI.

ANEXO 05

PROYECTO

<nombre del proyecto>

Versión 01

PLAN DE PRUEBAS

1. *Estrategia de prueba*
2. *Planificación de las pruebas*
3. *Matriz requerimientos / Casos de pruebas*
4. *Casos de prueba*
5. *Acta de Aceptación*

<día> de <mes> del <año>

1. Estrategia de Prueba:

Se define las estrategias de pruebas a utilizar para que el cliente de la aceptación a la aplicación.

- Objetivo y Alcance:

  Se debe indicar la estrategia de pruebas a utilizar dado que no siempre es posible probar el sistema completo, generalmente es una mezcla entre pruebas horizontales, verticales y de excepción.

- Riesgos
- Requisitos: Requerimientos de entornos como configuración técnica, interfaces externas, datos de prueba, automatización, etc.
- Definición de ambiente / entorno
- Requisitos de recursos humanos
- Criterios de termino de las pruebas

2. Planificación de las Pruebas

La ejecución de las pruebas y cada uno de los casos serán planificados por el área de Calidad del que estará a cargo de las ejecuciones.

3. Matriz Requerimientos / Casos de Prueba

Se presenta la matriz para vincular cada Caso de Prueba con el Requisito Funcional que se desea probar.

4. Casos de pruebas

Se lista los casos de prueba generados para el proyecto, cuyo detalle será especificado posteriormente en la matriz de Casos de Pruebas.

Caso de Prueba	Descripción del Caso de Prueba	Prueba de Opción
CP-01	Verificar el contenido del explorador de mantenimientos de maestros de Servicio Social.	Nuevo
CP-02	Realizar mantenimiento a Tipos de Visita: Insertar un nuevo tipo de visita	Nuevo
CP-03	Realizar mantenimiento a Tipos de Visita: Modificar un tipo de visita	Nuevo
CP-04	Realizar mantenimiento a Tipos de Visita: Eliminar un tipo de visita	Nuevo

BIBLIOGRAFÍA

REFERENCIAS BIBLIOGRÁFICAS

[1] D. Gelperin y W.C. Hetzel (1988). "The Growth of Software Testing". CACM 31(6). ISSN 0001-0782.

[2] Roger S. Pressman (2002)."Ingeniería de Software, Enfoque Práctico".7ma edición.

[3] G.J. Myers (1979)," The Art of Software Testing". John Wiley & Sons,Inc.

[4] Rubby Casallas, Darío Correal, Nicolás López. "Mejoramiento del Proceso de Pruebas en un Contexto de Desarrollo de Software Globalizado",11 págs.

[5] Liliana González Palacio, "MÉTODO PARA GENERAR CASOS DE PRUEBA FUNCIONAL EN EL DESARROLLO DE SOFTWARE", Revista Ingenierías Universidad de Medellín. vol.8 - 8págs.

[6] Elena Raja Prado, "Casi todas las pruebas del software". Actas de Talleres de Ingeniería del Software y Bases de Datos. Vol.1 – 4 págs.

[7] Edgar Serna M., Fernando Arango I. "Prueba del software: más que una fase en el ciclo de vida". Revista de Ingeniería. 7 págs.- diciembre 2011.

[8] Carlos M. Zapata, "Una Representación gráfica del testing ágil". Revista Avances en Sistemas e Informática. Vol.7- 10- Julio 2010.

[9] Ignacio Esmite, Mauricio Farías, Nicolás Farías, Beatriz Pérez." Automatización y Gestión de las Pruebas Funcionales usando Herramientas Open Source", 12 págs.

[10] JACOBSON, IVAR; G. BOOCH Y J. RUMBEAUGH: El proceso unificado de desarrollo de software. Rational SW. Corporation. Pearson Educación SA, Madrid, 2000.

[11] Dirección de Normalización - INACAL Ingeniería de software y sistemas. Procesos del ciclo de vida del software. NORMA TÉCNICA PERUANA NTP-ISO/IEC 12207.3ª Edición-2016.

[12] Macario Polo, Mario Piattini, Francisco Ruiz, Coral Calero: MANTEMA: a Software Maintenance Methodology Based on the ISO/IEC 12207 Standard' 1999- Escuela Superior de Informática Universidad de Castilla-La Mancha-España.

[13] Std. 610-1990 IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries - 18 Jan. 1991.

PÁGINAS WEB

[WEB01] https://sites.google.com/site/aseguramientodelacalidadidmf/marco-teorico/pruebas-y-teoria-de-pruebas

[WEB02] http://www.calidadysoftware.com/

[WEB03] http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/381

[WEB04] https://www.globetesting.com/2012/03/comparativa-de-herramientas-para-pruebas-automaticas/

[WEB05] http://raknarrok.blogspot.pe/